Cross Link Insertion for Variation Driven Clock Network Construction

QIAN, Fuqiang

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
August 2012

Abstract

Clock skew caused by variation is one of the most important problems in clock network synthesis today. Even if a clock network is designed to have zero skew, variation such as capacitive load and power supply will cause differences in arrival time of a clock signal. Non-tree clock network is considered to be an effective way to address the skew variation problem. Due to its inherent redundancy, clock mesh is very tolerant to variation. However, it costs much excessive amount of power compared to a clock tree. Link based non-tree clock network is an economic way to reduce clock skew caused by variation. Instead of using a dense mesh, only a number of links are inserted into a tree, so the power increase is small. Several existing works focus on the effect of cross link as well as the construction of such cross link structure. However, it is still not very clear where cross links should be inserted to achieve the most clock skew reduction with small wire resources. In this thesis, we propose a new method using linear program to solve this problem. In our approach, clock skew in a non-tree clock network is computed using an idea of load redistribution and non-tree decomposition. The delay information obtained is then used to select the node pairs for cross link insertion. Our methodology tries to insert cross links where skew can be reduced most effectively. Our method also considers tradeoff between cross link length and skew reduction effect. We compare our result with the most similar work on this problem [1] and a recent work

[4] which inserts links between internal nodes of a tree. Experiments show that our method can reduce skew under variation effectively. We achieve 28% clock skew reduction with only 40% link resources.

Acknowledgement

At the very beginning, I am deeply indebted to my supervisor, Professor Fung Yu Young, who patiently motivated me to conceive and develop the main ideas in the thesis and taught me so much in my research. I couldn't achieve these academic works without your supervision. I would like to express to her my sincere gratitude for her seasoned guidance from the very early stage of this research work as well as providing constructive advices throughout the entire study. In particular, I also would like to thank her and her family for their concerns about my daily life. Professor Yu Liang Wu and Profrssor Qiang Xu, thank you for your help and suggestion in my research work.

My research partners Linfu Xiao, Haitong Tian, thank you for your insightful comments on my research work. I am also grateful to all the colleagues, Tao Huang, Xu He, Guxin Cui, Yan Jiang, Yuan Jiang, Ka Chun Lam, it is you who bring me a colorful postgraduate study life.

Last but not the least, my family and my friends, without your love and support, I cannot achieve anything. I would like to give my greatest appreciation to you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Clock Distribution Network

In a synchronous digital design, a clock distribution network is used to order the events happening in a circuit. The distribution of clock signal has a strong implication on performance and power consumption. One of the most important problems in clock distribution network is clock skew, which is the difference in arrival times of clock signal at sinks as show in Fig.1.1. Given two different points i and j on an integrated circuit, the clock skew is given by $t_{skew} = t_i - t_j$ where t_i and t_j are clock arrival times at these two points. Ideally, clock paths from source to the sinks are going to be equal. Under ideal conditions, there are two basic constraints in a synchronous structure, one is

$$T > t_{c-q} + t_{logic} + t_{su}$$

where t_{c-q} is the maximum propagation delay of a register, t_{logic} is the maximum delay of the combinational logic between the two registers, t_{su} is the setup time for the registers. The clock period T should be long enough for the signal to propagate through the register and the logic and be set up at

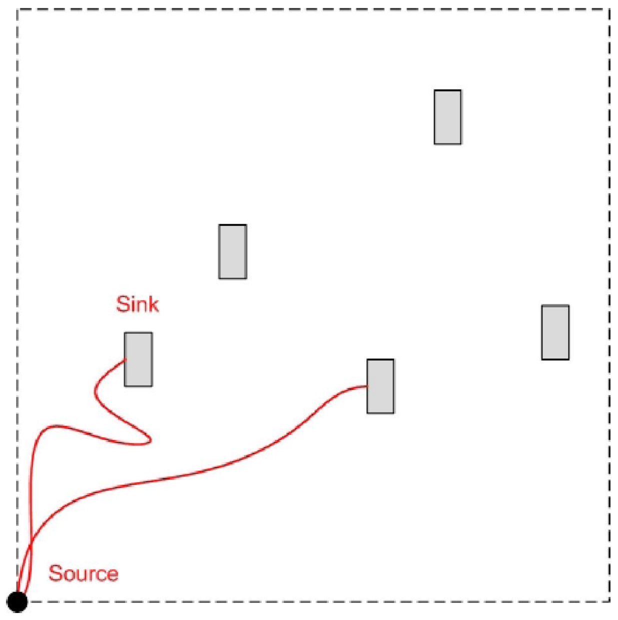


Figure 1.1: Difference in arrival times of clock signal

the destination register before the start of the next clock signal. On the other hand, the hold time constraint is

$$t_{hold} < t_{c-q,cd} + t_{logic,cd}$$

where $t_{c-q,cd}$ is contamination (minimum) delay through a register, $t_{logic,cd}$ is the minimum delay of the combinational logic. The hold time t_{hold} of the destination register should be shorter than the time required to propagate though the register and the logic.

However, clock paths are not perfectly equal, which results in clock skew. Clock skew can be positive or negative depending on the routing details. Non-zero clock skew has a significant impact on the functionality and performance of a sequential system. For example, in Fig.1.2, if t_{CLK2} is delayed by a positive t_{skew}, the clock skew will affect the clock period as

$$T > t_{c-q} + t_{logic} + t_{su} - t_{skew}$$

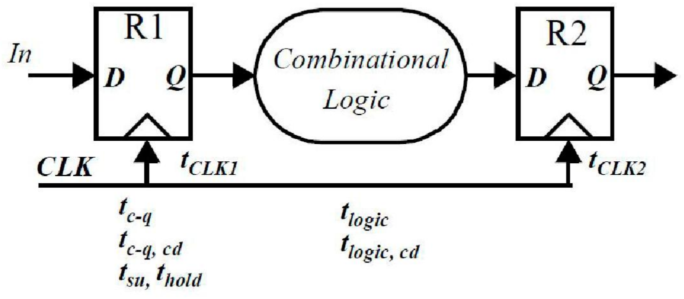


Figure 1.2: Basic structure of pipelined datapath circuit [22]

In this scenario, as $t_{skew} > 0$, the equation indicates that clock skew reduces the clock period and therefore improve the performance. However, the increased skew may harm the correct operation of the circuit due to race conditions. At the same time, negative clock skew will increase the clock period directly and can fundamentally limit the performance. In general, designing a clock distribution network with low skew is essential.

In the past years, efforts have been made towards achieving zero clock skew while reducing the total wire length [26] [27] [16] [17] [18] [19]. Some works such as [2] [33] explored the problem of non-zero useful skew routing. However, even if a clock distribution network is designed to have zero skew, process variation and environmental variation such as changes in capacitive load, power supply and temperature will influence the delivery of the clock signal. Sources of clock skew can be,

1. Manufacturing Variation. Clock buffer delay variation is critical to minimizing clock skew. Device parameters of the buffers vary along different paths, which results in clock skew. This is due to many sources of variations like dopant variation (that affects the depth of junctions and cause variation in threshold and parasitic parameters), oxide variation, etc.

2. Interconnect Variation. Interconnect resistance and capacitance vary

across different paths. There is deviation in the width of a wire and line spacing, which results in clock skew.

3. Environment Variation. Temperature is time varying and is usually considered to be a skew component. Power supply variation also limits the performance of clock networks.

4. Device Aging. Device aging is a result of the degradation of the gate dielectric. It degrades device characteristics and circuit performance over time.

Some of these variations can be modeled while some can not. Designing a robust clock network is important to handle the skew variation problem. Clock skew caused by variation is now one of the most critical problems that the design of large and high performance system is facing today.

One common structure to route a clock network is tree. The H-tree algorithm delivers signal from a central point to various points using balanced paths. An example of such structure is shown in Fig.1.3. The H-tree algorithm is applicable for regular array networks. A more general algorithm is RC matched clock distribution [20]. The interconnections delivering clock signal are of equal length. This general approach does not rely on a regular structure compared with the H-tree structure.

Mesh

Non-tree clock network is considered to be a promising way to address the variation problem. An example of a clock mesh is shown in Fig.1.4. The top level is a clock tree and a mesh is attached to the bottom of that tree. Clock sinks are ususally connected to the mesh edges. This clock mesh structure, because of its inherent redundancy, is more tolerant to process variation

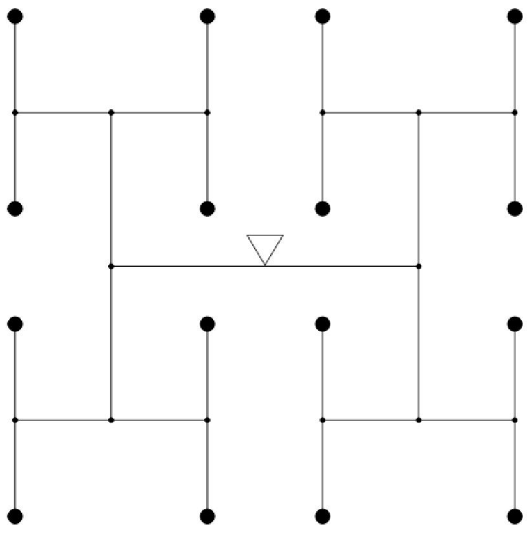


Figure 1.3: H-tree

and is able to provide lower skew variability compared with traditional clock tree. However, clock mesh has the disadvantage of causing much more power dissipation because of the longer wire length. This type of clock distribution network is used on several microprocessor chips, achieving very low skew [5]. A sliding window based scheme is used to analyze a clock mesh [7]. A combinatorial algorithm to optimize a clock mesh is proposed in [6]. A comprehensive and automated framework for planning, synthesizing and optimization of clock mesh networks is proposed in [8]. A mixed tree-mesh clock network achieving low skew is presented in [25].

Cross Link

Compared to the mesh structure, clock network constructed by inserting cross links consumes much less power. An example of clock tree with cross links is illustrated in Fig.1.5. Instead of using a dense mesh, only a number of cross links are inserted into a clock tree. This structure combines the advantages of clock tree and clock mesh. A framework to construct such a non-tree

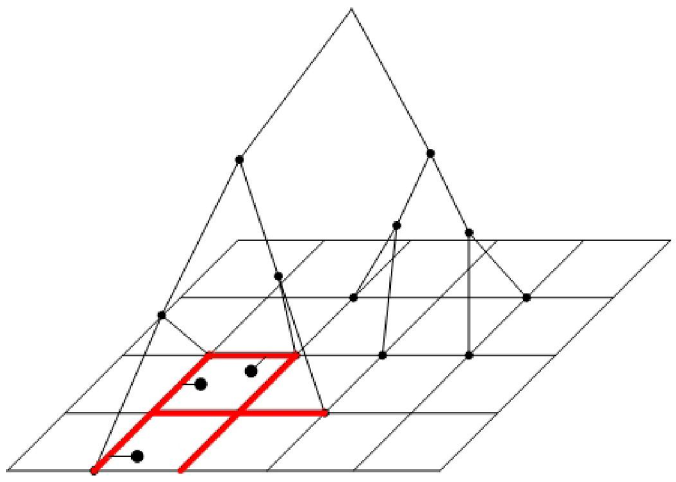


Figure 1.4: Clock Mesh

network with some analysis on the effect of link insertion on skew variability is first proposed in [1]. A statistical based non-tree clock network construction technique is presented in [9]. Later, the cross link idea is extended to handle buffered clock tree in [10] [11]. Wire sizing is performed to improve skew variability in a non-tree topology in [12]. Recently, a link insertion scheme that inserts cross links at higher level internal nodes in a clock tree, instead of sink nodes as in many previous works, is proposed in [4]. A cross link insertion scheme using linear program to select node pairs for link insertion is proposed in [28].

1.2 Our Contributions

Although there are a number of research works on non-tree clock network construction with cross links, some important questions are still unanswered. Most existing works on link insertion attempt to reduce skew variability while using shortest wire length. However, these approaches did not consider delay and skew when inserting links, which limits the effectiveness of link insertion.

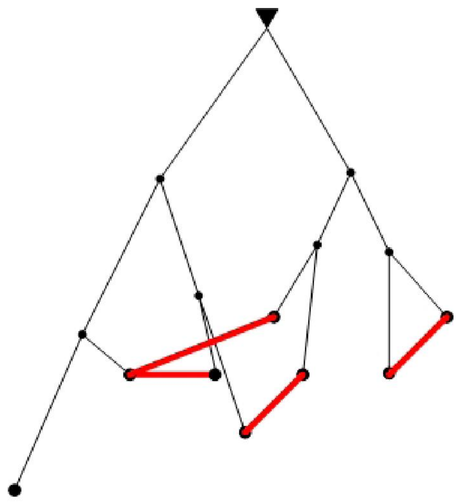


Figure 1.5: Cross Link

The tradeoff between the length of a cross link and its ability to reduce clock skew is not analytically studied. In this thesis, efforts are made towards solving these problems. Our contributions can be summarized as follows.

- We use the idea of load redistribution and tree decomposition [4] to obtain the delay and skew information in a non-tree clock network. We study where cross links should be inserted into a clock network.
- We formulate the cross link insertion problem as a sequence of linear programs, with an objective to find a pair of nodes to insert a cross link such that the skew variability can be reduced the most while the wire length increase due to the link insertion is constrained. We considers tradeoff between cross link length and skew reduction effect. By applying this technique recursively, we can add a user-defined number of cross links and the clock skew will be reduced progressively.
- Simulation results show that our method can lead to significant skew reduction under variations.

1.3 Organization of the Thesis

The thesis is organized as follows. We first review the literature in clock network construction in Chapter 2, in which we focus on the most recent research dealing with skew variability. In Chapter 3, techniques for computing signal delay and clock skew in non-tree clock networks will be studied. Our method to construct a non-tree clock network with cross links will be presented. We formulate the node pair selection problem as a sequence of linear program. The result is used to select the node pairs for cross link insertion where skew can be reduced most effectively. In Chapter 4, we discuss cross link insertion in buffered clock network. Finally, a summary of our work will be included in Chapter 5.

□ **End of chapter.**

Chapter 2

Literature Review

In this charpter, we will review the literature in clock network construction. Firstly, some important papers on zero skew clock routing while minimizing the total wire length will be presented. Then we will discuss several papers on the synthesis and optimization of clock mesh network. Finally, we will reivew some recent papers on clock network construction with cross links.

2.1 Exact Zero Skew

In the paper [16], an exact zero skew clock routing algorithm is presented based on the Elmore delay model rather that just wire length balancing. Zero skewed subtrees are interconnected to a new zero skew tree recursively based on the delay computation. The recursive bottom up approach is applied to construct a complete zero skew tree.

Firstly, a linear time delay computation method is studied. A clock tree is modeled as a RC tree. Each branch is associated with a resistance value and each node is associated with a capacitive value. The capacitance of a node can be computed from its own node capacitance and the capacitances of its successors. The delay time can be computed from the delay of its predecessor, the node capacitance and the branch resistance.

Based on the delay model and the delay computation method, a recursive bottom up zero skew algorithm is presented. By tuning the tapping point, any two zero skew subtrees can be merged to form a new zero skew tree. Repeating this process until reaching the root will generate a complete tree. In some cases, wire elongation is needed to balance two subtrees. More details of the algorithm are discussed blow.

Consider the case in whch we want to merge the roots of two subtrees. In order to have zero skew merging, the following equality should hold:

$$r_1(c_1/2 + C_1) + t_1 = r_2(c_2/2 + C_2) + t_2$$

where t_i refers to delay from node i to the sinks, r_1 and c_1 (r_2 and c_2) are resistance and capacitance of wire 1 (wire 2). If we assume that the total wire length connecting the two nodes is l, and the length of wire 1 is xl. Hence, the length of wire 2 is $(1-x)l$. Let α and β be the resistance and capacitance per unit length of wire respectively. Then $r = \alpha l$, $r_1 = \alpha xl$, $r_2 = \alpha(1-x)l$, $c = \beta l$, $c_1 = \beta xl$, $c_2 = \beta(1-x)l$. After solving the above equation, we have

$$x = \frac{(t_2 - t_1) + \alpha l(C_2 + \beta l/2)}{\alpha l(\beta l + C_1 + C_2)}$$

If $0 \leq x \leq 1$, the tapping point is on the line segment interconnecting the roots of the two subtrees. In case if $x \leq 0$ or $x \geq 1$, extra wire is needed to achieve delay balance. For example, if $x \leq 0$, the tapping point has to be on the root of subtree 1. Assume that the length of the elongated wire is l', the resistance of this wire is $\alpha l'$, and its capacitance is $\beta l'$. To balance delay, it requires

$$t_1 = t_2 + \alpha l'(C_2 + \beta l'/2)$$

or l' is given by

$$l' = \frac{(\sqrt{(\alpha C_2)^2 + 2\alpha\beta(t_1 - t_2)} - \alpha C_2)}{\alpha\beta}$$

Similarly, we can determine l' if $x \geq 1$.

The algorithm is tested on five different sized benchmarks. It is observed from the results that the exact zero skew algorithm performs better than the wire length balancing algorithm with zero skew and smaller phase delay.

2.2 DME Algorithm

In the papers [17] [18] [19], an algorithm called Deferred-Merge Embedding (DME) is proposed. The algorithm constructs a zero skew clock network while reducing the total wire length. It contains a bottom up merging phase and a top down embedding phase. The bottom up merging phase computes the possible loci of the internal nodes of a tree. Then a top down embedding phase determines the exact location of each tree node.

A Manhattan arc is a line segment with slope +1 or -1. The collection of points within a fixed distance of a Manhattan arc is called a tilted rectangular region (TRR). The boundary of a TRR is composed of Manhattan arcs. The core is the Manhattan arc at the center of a TRR. The radius of a TRR is the distance between its core and its boundary. In the bottom up merging phase, a tree of merging segments is constructed given a connection topology G. A merging segment $ms(v)$ of node v represents the possible placements $pl(v)$ of node v. The merging segment of node v depends on its children a and b. If v is a sink s_i, then $ms(v) = s_i$. If v is an internal node, $ms(v)$ is obtained by intersecting two TRRs, trr_a with core $ms(a)$ and radius $|e_a|$ and trr_b with core $ms(b)$ and radius $|e_b|$.

Once the tree of segments has been constructed, the exact embeddings of the internal nodes are chosen in a top down phase,

1. If v is the root node, select any point in $ms(v)$

2. If v is an internal node, choose any point in $ms(v)$ that is within distance $|e_v|$ from the placement of v's parent p

In summary, the Deferred-Merge Embedding (DME) algorithm was proposed to embed any given connection topology to create a clock tree with zero skew while minimizing total wire length. Experimental results show an 8% to 15% wire length reduction over previous methods in [16]. The DME algorithm can be used with both the Elmore and linear delay model. The algorithm is very fast, running in linear time of the number of sinks.

2.3 Combinatorial Algorithms for Fast Clock Mesh Optimization

A mesh offers high tolerance to process variation at the cost of high power consumption. The authors of [6] offers designers a tradeoff between power and clock skew reduction. A set-cover formulation is used to obtain the minimum buffer resource to drive a mesh under slew constraints. After obtaining a buffered mesh, based on the survivable network theory, only those critical edges are retained to reduce the power consumption of the clock mesh.

The mesh reduction method is as follows. Let $m \times n$ denotes the dimension of the clock mesh M. I denotes the set of nodes in the mesh. $S = \{s_1, s_2 \ldots s_n\}$ denotes the set of sinks in M. Each sink s_i is connected to a node $i \in I$. d_{ij} denotes the minimum distance between node i and node j in

the mesh. When some edges are removed from the mesh, the results should satisfy:
Each sink s_i is connected to at least k nodes and for each such node j

1. $d_{ij} \leq L_{max}$.
2. There are at least l paths between node i and node j.
3. There is at least a buffer on such node j.

Parameters k, L_{max} and l are user defined. The algorithm will maximize the number of edges removed. The variables k and l control the tradeoff between tolerance to variation and power dissipation. If both k and l are large, the number of edges removed will be small, which means more redundancy and power dissipation.

The mesh reduction problem is transformed into a Steiner Network problem [29] [30] by

1. Representing the mesh by a graph $G = (V, E)$.
2. Setting the connectivity requirement function $r(u, v) = 0$ for all $(u, v) \in V$.
3. Finding k closest mesh buffer locations T_i for each sink s_i.
4. Setting $r(p, j) = l$, $\forall\ p \in$ all s_i and T_i, $\forall\ j$.

A greedy heuristic is used to solve the Steiner Network problem. The algorithm first initializes the cost of all edges to unity. Then edge disjoint paths between clock sinks and the closest k mesh buffers will be indentified.

In summary, the mesh reduction problem is formulated based on the survivable network theory. Then the mesh reduction problem is transformed into Steiner Network and solved by a greedy heuristic. The technique allows the designers to tradeoff between skew tolerance and power dissipation. The algorithm is fast and experimental results show that there are 29% reduction in wire length and 28% reduction in power.

2.4 MeshWorks: An Efficient Framework for Planning, Synthesis and Optimization of Clock Mesh Networks

In the paper [8], a comprehensive framework called MeshWorks is presented for planning, synthesis and optimization of clock mesh. They address the problem of obtaining an initial clock mesh. A simple method considering wire length and skew is used to choose a mesh size. An algorithm based on the network sensitivity theory is applied to select the mesh edges that can be removed to reduce the size of the clock mesh.

The mesh planning and synthesizing algorithm considers the relationships between the total wire length, clock skew and mesh size. For example, suppose a mesh is denoted by $m * n$ and the sinks $S = \{s_1, s_2 \ldots s_n\}$ are connected to mesh nodes with stubs. The wire length of the mesh is a linear function of the mesh size as illustrated in Fig.2.1. As m or n increases, the mesh will become more dense. Since each sink is associated with a stub, a denser mesh will result in smaller total stub length and the mesh wire length is larger than the stub wire length. However, for a sparse mesh, the mesh wire length will be less than the stub wire length. An appropriate mesh size can be found to minimize the total wire length, as in Fig.2.1.

Let del_i denote the delay of sink s_i and $w(p, q)$ denote the width of a mesh segment between node p and node q. The effect of removing a mesh segment

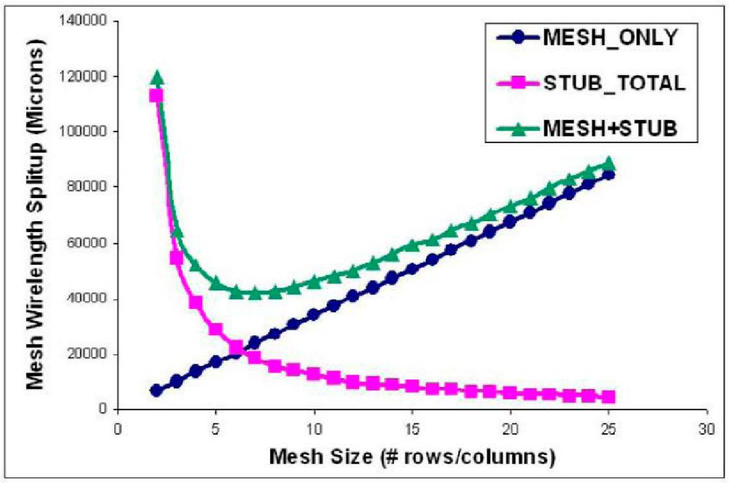


Figure 2.1: Determine mesh size [8]

between node p and node q is defined by:

$$Cost(p,q) = \underset{j,k \in sinks}{Max}\{\frac{\partial del_j}{\partial w(p,q)} - \frac{\partial del_k}{\partial w(p,q)}\} * w(p,q)$$

The algorithms try to remove the mesh edges with the lowest cost function to obtain a optimized mesh structure.

Details of the mesh optimization approach is as follows:

1. Cluster sinks close to each other.

2. Sinks in a cluster will be merged into a single sink with equal total capacitance. Therefore, an approximate mesh with fewer nodes can be obtained.

3. Obtain delay sensitivities of merged sinks with respect to the mesh edges. Based on the delay sensitivities, obtain the $cost(p,q)$ for each mesh edge $e(p,q)$. The cost function is the maximum clock skew change in the mesh when a certain edge is removed.

Experimental results indicate that their algorithms can achieve 26% reduction in buffer area, 19% reduction in wirelength and 18% reduction in

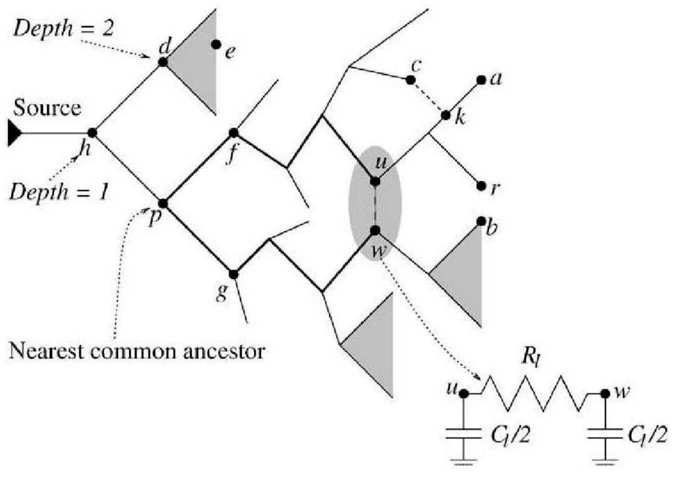


Figure 2.2: Inserting cross link [1]

power dissipation compared to a recent work [6]. A compressive framework for both mesh planning and optimization is proposed in this work.

2.5 Reducing Clock Skew variability via Cross Links

Rajaram *et al.* [1] propose a non-tree clock structure by inserting cross links into a tree. This non-tree structure consumes much less power compared to the traditional mesh structure. Some analysis of the effect of cross link is presented in their work. They also propose two schemes to generate a non-tree structure with cross links from a tree. These two schemes do not utilize delay information to insert cross links.

The impact of cross link is analyzed by a technique by Chan and Karplus [3]. Assume that a cross link is inserted between node u and w as illustrated in Fig.2.2. The delay at a node i in the clock tree after link insertion is given by

$$t_i{}' = t_i - \frac{t_u - t_w}{R_l + r_u + r_w} r_i$$

where t_i, t_u and t_w are the delays before link insertion, R_l is the link resistance, r_i, r_u and r_w is the delay at node i, u and w respectively when $C_u = 1, C_w = -1$ and other node capacitances are set to zero.

We assume that the skew between link endpoints node u and node w is $q_{u,w}$, the skew after link insertion is given by

$$q_{u,w}{}' = \frac{R_l}{R_l + r_u - r_w} q_{u,w}$$

It is claimed that a link resistor will reduce the skew variability of two zero skew link endpoints. However, the skew between two arbitrary nodes after link insertion is not very clear. Skew variability may be reduced or increased. On the other hand, the analysis and observations are only based on the first link added into the tree. Analysis of more links in the tree will be very difficult.

Based on the analysis, after obtaining a clock tree, their algorithm indentifies node pairs for link insertion. They propose two schemes for selecting node pairs for link insertion. The first one is a rule based selection scheme. This scheme is simply choosing all sink pairs satisfying three simple rules from the analysis. For example, the depth γ of the nearest common ancestor of a selected sink node pair needs to be no greater than a bound γ_{max}. Another scheme is a minimum weight matching based selection scheme. For different depths, a user defined number of links will be selected. For instance, $k = (2, 1, 1)$ indicates that there are two links for each subtree at depth $\gamma = 1$, and there is one link for each subtree at depth $\gamma = 2$ and $\gamma = 3$. The subproblem to select a node pair between two subtrees is modeled in a bipartite graph problem for selecting node pairs and can be solved by the minimum weight matching algorithm.

The algorithm is summarized as follows

1. Obtain an initial clock tree.

2. Select node pairs for link insertion.

3. Add link capacitance and tune tapping points.

4. Insert link resistance.

The experiments are performed on both mesh and cross link networks. A dense mesh usually performs better than cross link structure at the cost of significant wire increase. The minimum weight matching-based link method always outperforms the naive rule-based method on both skew and wire length minimization.

To summarize, a cross link structure is proposed in this paper to handle the skew variation problem. Some analysis of the effect of cross link is presented. Based on the analysis, two schemes which did not consider delay are proposed to construct clock network with cross links.

2.6 Statistical Based Link Insertion for Robust Clock Network Design

This paper presents a statistical based algorithm for clock network construction with cross links [9]. The process starts with a tree and incrementally insert cross links. The skew variation of the final clock network is within a certain confidence interval under variation in wire width. In their work, a fitted Elmore delay model is applied to the delay computation of clock trees and non-trees. They use statistical timing analysis to obtain the statistical distribution of the clock skew in the network and determine the optimal insertion point. Once a cross link is inserted, the statistical skew distribution is updated before more links are inserted.

The Elmore delay of a wire is expressed as a random variable under wire width variation. Once a link is added, RC is updated to obtain the delay value. The approach is similar to that in [1] as follows:

$$R_xC_x = [R_{x-1} + \frac{R_{x-1}V_xV_x{}^TR_{x-1}}{r_x - V_x{}^TR_{x-1}V_x}](C_{x-1} + \triangle C_x)$$

where $R_{x-1}C_{x-1}$ refers to the RC delay before link insertion. Hence the term can be found starting from R_0C_0. One difference from [1] is that they also add the link capacitance which is considered in the term $R_0 \triangle C_x$.

For each candidate pair of nodes, a link is temporarily inserted. The statistical delay and skew after link insertion is computed to decide if the candidate link should be inserted permanently. Once a link is inserted, delay and skew is updated for the next link insertion. There is a limit on the link distance. The reason is to reduce the search space to save run time. Another reason is that a long wire length consumes more wire resource and has a large perturbation on the original clock network. The other constraint is the height of the nearest common ancestor of the link endpoints. The taller the subtree, the larger the effect the link has.

The flow can be summarized as follows

1. For each sink pair considered for link insertion, find the worst skew violation

2. Insert the link has the best worst case violation

3. Update skew

In the experiments, the skews are evaluated based on a normal distribution of the wire width. Monte Carlo simulations are performed on the original clock tree and the link-inserted tree. To summarize, their approach is only able to handle wire variation which effects clock skew. The analysis is complicated and the runtime is very long even for a benchmark with several hundred sinks.

2.7 Variation Tolerant Buffered Clock Network Synthesis with Cross Links

The work [11] deals with the problem of constructing a link based buffered clock network. Their approach makes use of ordinary buffer and does not require SPICE for tuning the location of the tree nodes. A balanced buffered clock network with cross links is constructed which is tolerant to variation.

Their approach requires a link insertion friendly clock tree (a balanced buffered clock tree) which is more tolerant to variation. A merging scheme to construct a balanced buffered clock tree while simultaneously trying to minimize the total wire length is introduced. The general framework is similar to the DME algorithm. In their approach, the slew information is propagated in a bottom up fashion. When merging a given pair of subtrees, the subtree capacitance and slew information will also be obtained. Subtrees are merged when the capacitance after merging is less than the capacitance limit. Buffers are inserted at the roots of all the subtrees when there is no node pair merging satifying the capacitance limit. Their approach results in a balanced tree and the numbers of buffers from the source to every sink are the same.

The key merging step is described below. They maintain two separate lists F for flagged nodes and U for unflagged nodes. The nodes in U are considered for link node pair selection. For a node i in U, if a suitable node pair for merging without exceeding the effective downstream capacitance limit is not found, this node will be removed from U and added to the list F. The node pair selection process will be repeated until the list U is finished. Buffered will be added to the unmerged nodes in F. Delays, slews and downstream capacitance will be updated and the nodes in F are moved back to list U. The process will continue until a complete clock tree is obtained.

After a balanced clock tree is constructed, sink node pairs will be selected for link insertion by a modified MST algorithm. The cost function is a weighted function of delay and link length. This top level algorithm is similar to [1]. The steps can be summarized as follows:

1. Construct a balanced buffered clock tree using the techniques discussed above.

2. Select node pairs for link insertion using a modified MST algorithm.

3. Add extra link capacitance and construct buffered clock tree with the same topology.

4. Insert link to obtain the final clock network.

In the experiments, their buffered clock network with links has the best skew reduction compared with other algorithms. The percentage for extra link length is small. Runtimes are much less that those algorithms using SPICE to tune the locations of the tree internal nodes.

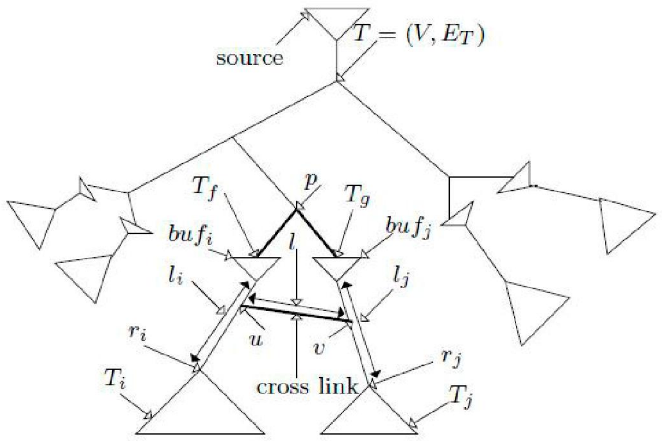


Figure 2.3: Cross link between internal nodes [4]

2.8 Cross Link Insertion for Improving Tolerance to Variations in Clock Network Synthesis

Instead of inserting links between sinks, the authors of [4] propose a scheme to insert links between internal nodes of a clock tree. An example is shown in Fig.2.3. In this example, a cross link is inserted between two zero skew internal nodes u and v below buffers rather than inserted between sink nodes. The link insertion process is integrated into the tree construction process. Both the clock skew variability and the total link length are reduced.

The scheme comprises of merging, buffer insertion and link insertion. The high level framework is similar to the DME approach. Nearest neighbor graph is constructed using root nodes of subtrees as vertices. When two nodes are merged to form a new subtree, slew check is performed to decide if buffers should be inserted on top of the two nodes using stem wire. The stem wire length is found by binary search using NGSPICE. Each buffer buf_i

has a merging region mr_{buf_i} associated with it. The problem of blockage is considered by chopping off the part of a merging region lying within a blockage.

It is claimed in [1] that a link is beneficial only when it is inserted between two zero skew nodes. So they propose to insert a link between two zero skew internal nodes on the stem wire. The location search process is also guided by NGSPICE simulations. The proposed link insertion flow helps to control link length and reduce the total clock network capacitance. Cross link helps improving the correlation of sink delays between the two subtrees where cross links are inserted. Links are inserted below buffers, which helps to reduce variation effects of buffers.

Simulation results on the ISPD 2010 contest benchmark demonstrate the effective of their approach. In the experiments, they used a 15% supply voltage variation and a 10% wire width variation. The buffers used consists of 10 parallel buffers driving 40 parallel buffers (10 buffers in the first layer, 40 buffers in the second layer). The proposed scheme obtains less capacitance compared with the top three teams of the ISPD contest while satisfying the local clock skew constraints.

□ **End of chapter.**

Chapter 3

Clock Network Construction with Cross Links

3.1 Signal Delay and Clock Skew in Non-tree Clock Network

We will first review the delay calculation technique in general RC network [13]. Using this technique, a non-tree clock network can be decomposed into a tree with some way of load redistribution. Signal delay in the decomposed tree is exact the same as that in the original non-tree network. Therefore, finding delay in a non-tree network becomes simply calculating Elmore delay in a RC tree. Secondly, we will study the effect of a cross link from the perspective of load redistribution. We deduce and verify that the skew reduction effect of a cross link is affected by the resistance of the link R_{link}. If a link resistance is large, its skew reduction effect will be small. So a cross link with small resistance has more potential to improve skew performance. This observation can be used to guide our link insertion process.

3.1.1 Computing Delay in Non-tree Network

Calculation of the delays in a non-tree network is non-trivial, because there are loops between the nodes, which makes possible that one node is driving and loading another node at the same time. The relationships between nodes are not explicit in a non-tree clock network. In the past, several techniques have been suggested to model the delay in a general RC network [3] [13] [14].

Based on the Elmore delay model, a technique called tree decomposition and load redistribution is proposed in [13]. Their approach can be used to calculate the delays of the nodes in a general RC network with resistance loops. Suppose that N is an arbitrary node in a general RC network and N has k neighboring nodes, denoted by M_i where $i = 1, \ldots, k$. Node N is connected to M_i through edge E_i with a resistance of value R_i. Denote the load capacitance of N as C. The idea is to partition and distribute C into N's k neighboring edges. Let C_i be the equivalent load distributed to edge E_i, where C_i can be positive or negative. According to Elmore delay, if the delay of node M_i is T_i, the delay of node N will be $T = T_i + R_iC_i$. Then there is a set of constraints as follows:

$$\sum_{i=1}^{k} C_i = C \tag{3.1}$$

$$T = T_i + R_iC_i \qquad i = 1, \ldots, k \tag{3.2}$$

With the above constraints, a general RC network can be decomposed into a tree with some way of load redistribution. Signal delay in the decomposed tree is exact the same as delay in the original non-tree. Since the delays in a tree can be determined efficiently, we are able to calculate the delay in the original RC network.

Consider the effect of inserting a link between two nodes N_i and N_j with delays D_i and D_j respectively in a tree as shown in Fig.3.1. Assume that the

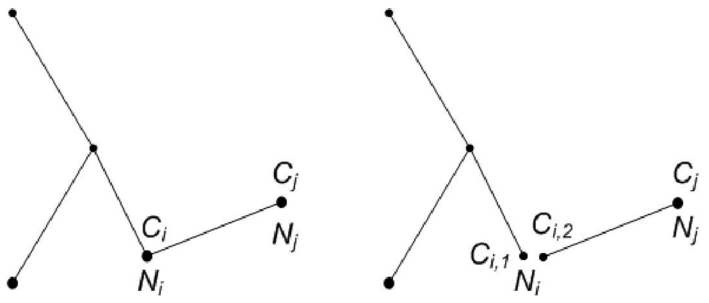


Figure 3.1: Load Redistribution through a Link

load capacitances of node N_i and N_j are C_i and C_j respectively before link insertion. After a link is inserted between node N_i and N_j, node N_j becomes one of the neighboring nodes of N_i. The loading capacitance of node N_i will be redistributed as in Fig.3.1. In this example, the load capacitance of node N_i is redistributed. Note that we can also split N_j instead of N_i. Suppose $C_{i,1}$ is the capacitance remaining at node N_i and $C_{i,2}$ is the load redistributed from node N_i to node N_j through the inserted link. Let D_i and D_j be the new delays at node N_i and N_j respectively, which can be calculated after this decomposing step.

According the set of constraints (3.1) and (3.2), we have the following equality constraints for this particular example:

$$C_{i,1} + C_{i,2} = C_i \tag{3.3}$$

$$D_i{}' = D_j{}' + D_{link} \tag{3.4}$$

The delay difference between node N_i and node N_j becomes $D_{link} = R_{link}C_{i,2}$. Note that if $C_{i,2}$ is positive, it indicates that node N_j is actually driving node N_i after this link insertion step and node N_j load part of node N_is capacitance, which is $C_{i,2}$. If $C_{i,2}$ is negative, it indicates that node N_i is driving node N_j instead.

3.1.2 Effect of a Cross Link on Clock Skew

The Elmore delay of node N_i in a tree T is

$$\sum_k R_{i,k}C_k \tag{3.5}$$

where $k \in$ all nodes in T, $R_{i,k}$ is the shared path resistance between node N_i and node N_k and C_k is the capacitance of node N_k. Consider the same example in Fig.3.1, a link is inserted between node N_i and node N_j. For a given tree, the resistances of edges and capacitances of the nodes are known, so the delays at nodes N_i and N_j can be expressed by C_i and C_j as follows with constants k_i, where $i = 1, 2, \ldots, 6$:

$$D_i = k_1C_i + k_2C_j + k_3 \tag{3.6}$$

$$D_j = k_4C_i + k_5C_j + k_6 \tag{3.7}$$

The skew between node N_i and N_j before link insertion is

$$q_{ij} = D_i - D_j \tag{3.8}$$

After a link is inserted between N_i and N_j as in Fig.3.1, the delays become

$$D_i{}' = k_1(C_i - C_{i,2}) + k_2(C_j + C_{i,2}) + k_3 \tag{3.9}$$

$$D_j{}' = k_4(C_i - C_{i,2}) + k_5(C_j + C_{i,2}) + k_6 \tag{3.10}$$

The skew becomes $q_{ij}{}' = D_i{}' - D_j{}'$, which is given by

$$q_{ij}{}' = q_{ij} - (k_1 - k_4)C_{i,2} + (k_2 - k_5)C_{i,2} \tag{3.11}$$

which equals $R_{link}C_{i,2}$, so we have

$$C_{i,2} = \frac{q_{ij}}{k_1 - k_2 - k_4 + k_5 + R_{link}} \tag{3.12}$$

After solving $C_{i,2}$, the skew after link insertion can be written as

$$q_{ij}{}' = R_{link}C_{i,2} = \frac{R_{link}q_{ij}}{k_1 - k_2 - k_4 + k_5 + R_{link}} \tag{3.13}$$

The skew after link insertion is scaled by a factor $\frac{R_{link}}{k_1-k_2-k_4+k_5+R_{link}}$ compared with the skew before link insertion. As k_1, k_2, k_4, k_5 are constants, the skew reduction effect is determined by the resistance of the cross link R_{link}. If the link resistance is large, its skew reduction effect will be small. A link with smaller resistance has a higher potential to improve the skew performance. Although the above analysis is based on a structure that is a tree, the same argument can be applied even after several links are inserted since a non-tree network can be represented by a tree structure by applying the above load redistribution method.

3.2 Link Insertion for Non-tree Clock Network

Based on the techniques discussed in section 3.1.1, we are able to calculate signal delays in a clock network. In our approach, the sink load capacitances are modeled as variables and each node's Elmore delay is written as a function of the sink capacitances and the redistributed load capacitance. Each sink node delay will change according to the values of these capacitances. Therefore, there is a worst case skew under a sink capacitance variation. The worst case skew is defined as the maximum delay difference that might appear in the clock network under the variation of the sink capacitance. It is used as a metric to evaluate the clock skew performance in our work.

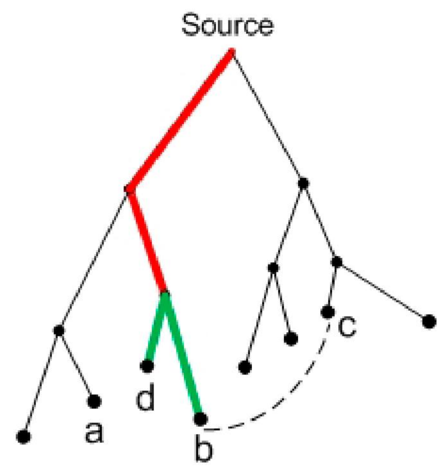


Figure 3.2: Clock Network with Links

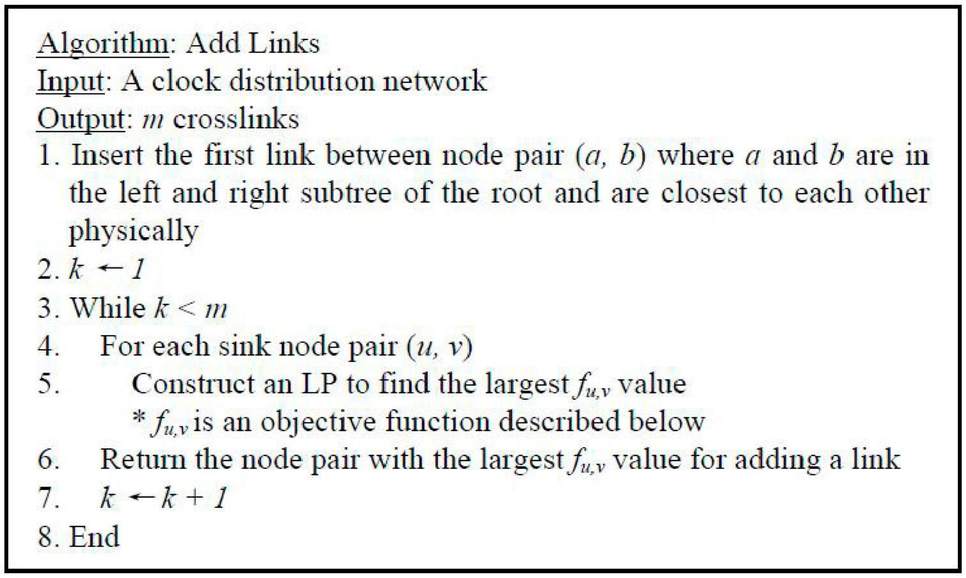
3.2.1 Motivation of Computing Delay for Link Insertion

Adding links between the node pairs which are most susceptible to have the worst case clock skew will be beneficial. Therefore, one important step is to identify the node pairs which will have large skew under variation. A shared path length l_{ij} is the length of the path shared between the path from the root s to sink node S_i and the path from s to S_j. It is obvious that if two nodes share a long common path from the source in a tree topology, the delay difference will be small. If the shared path length l_{ij} is small or even zero, the delay difference between S_i and S_j can be high. Those node pairs with small shared path length are topologically far away. In a clock tree, adding a link to those topologically far away node pairs will be beneficial because they are likely to exhibit a large clock skew. For example, in Fig.3.2, adding a link between node b and node c will generally be better than adding a link between node b and node d, because node pair (b, d) is less likely to have a large skew.

When the clock network is no longer a tree structure after some cross links are inserted, the relationship between nodes becomes unexplicit. It is difficult to tell if a node pair is topologically far away or not. In Fig.3.2, if we consider node pair (a, b) and node pair (d, c), node pair (a, b) are topologically closer if no links are inserted, since the lowest common ancestor (LCA) of (a, b) is at level 1 while the LCA of (d, c) is at the root (level 0). However, if a link is inserted between node b and node c, it is not possible to tell which node pair are topologically closer thus less likely to exhibit the worse skew without calculation. Therefore, we will use the non-tree delay calculation technique (Section 3.1.1) to obtain the Elmore delay and the worst case skew of each sink node pair under variation. Then we can decide the node pairs which have the highest potential to reduce clock skew after link insertion between them.

3.2.2 Overall Flow for Cross Link Insertion

In our approach, we will insert the first link between the two subtrees of the root in a clock tree network. We will find the node pair with the smallest physical distance and thus the smallest link resistance. Starting from the second link, we will formulate a sequence of linear programs to find the most beneficial node pair for inserting a link. Fig.3.3 shows an overall flow of our algorithm to insert links. For each sink node pair considered in link insertion, we find the maximum under capacitance variation. A link will be added to the node pair with the maximum possible skew. The linear program is then updated to find the next node pair for the third link insertion. In the following section, we will discuss in more details of the linear program.



```
Algorithm: Add Links
Input: A clock distribution network
Output: $m$ crosslinks
1. Insert the first link between node pair ($a$, $b$) where $a$ and $b$ are in
   the left and right subtree of the root and are closest to each other
   physically
2. $k \leftarrow 1$
3. While $k < m$
4.     For each sink node pair ($u$, $v$)
5.         Construct an LP to find the largest $f_{u,v}$ value
           * $f_{u,v}$ is an objective function described below
6.     Return the node pair with the largest $f_{u,v}$ value for adding a link
7.     $k \leftarrow k + 1$
8. End
```

Figure 3.3: Link Insertion Overview

3.2.3 Linear Program for Selecting Node Pairs

Given a non-tree clock network, we will first make use of the techniques discussed in Section 3.1.1 to decompose a network into a tree structure and redistribute the load capacitances. An example is illustrated in Fig.3.4. In Fig.3.4, a link k is added between sink nodes S_1 and S_2, with load capacitances C_1 and C_2 respectively. First we add half of the link capacitance C_{link} to endpoints S_1 and S_2 of the link. When the tree is decomposed, we pick one of the link endpoints to decompose. In this example, we pick node S_1 to be viewed as being split into two nodes as in Fig.3.4. The new node S_3 is connected to S_2 through the added link. The new loading capacitance at node S_1 is $C_1 + C_{link}/2 - c_k$ where c_k is the capacitance distributed to node S_2. Suppose that there are m links in the clock distribution network, similar decomposition can be done for all the links. We can use m variables $c_1, c_2, \ldots, c_m$ to describe the load redistribution from one endpoint of a link

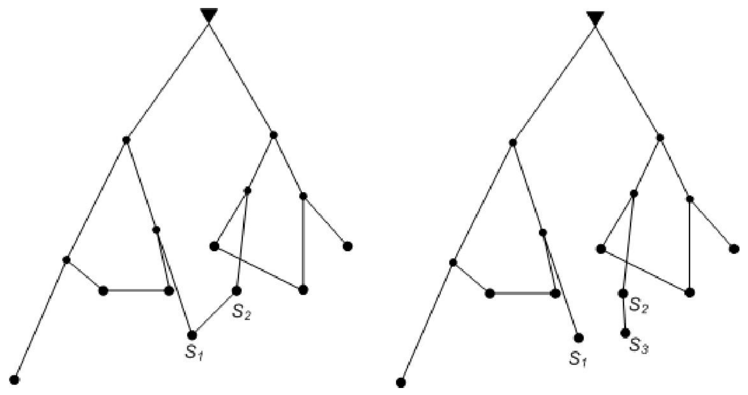


Figure 3.4: Decomposition of Clock Network with Links

to the other. Finally, we obtain a tree structure with some way of load distribution. Node's delay in the decomposed tree is exact the same as that in the original non-tree network.

Once the network is decomposed into trees and the load capacitances are redistributed, according to equation (3.5), we can write the Elmore delay of each sink node as a linear function of the sink load capacitances. Let n be the total number of sinks and m be the number of links, the delay of a sink node S_u is a linear function of the sink load capacitances $C_1, C_2, \ldots, C_n$ and the redistributed load capacitance $c_1, c_2, \ldots, c_m$,

$$t_u = f_u(C_1, C_2, \ldots, C_n, c_1, c_2, \ldots, c_m) \tag{3.14}$$

The skew between sink node S_u and S_v is

$$q_{u,v} = |t_u - t_v| \tag{3.15}$$

Then we can formulate the worst case skew between a pair of nodes under capacitance variation. However, adding a link between the pair with the worst skew might result in a significant increase in link length. If the length of the link is large, the resistance of the link will be high. Firstly, this will make the clock network have very long route and thus not practical. Secondly, a long wire will have lower skew reduction effect as discussed in Section 3.1.2. For instance, in Fig.3.2, if we do not consider the resistance of the link, adding a link between node b and node c will be better than adding a link between node a and node b, since node a and node b share some parts of their paths from the source and thus the delay difference will be smaller. However, if the physical distance and thus the link resistance between node b and node c is larger than that between node a and node b, it will be difficult to tell which link is better. Therefore a tradeoff between the length of a link and its clock skew reduction effect is needed. When selecting node pairs, we consider both clock skew and link length to find the node pairs which are most beneficial for link insertion. We thus define an objective function to strike a balance between the worst case skew and the link resistance $r_{u,v}$ as follows:

$$f_{u,v} = \frac{q_{u,v}}{r_{u,v}} \tag{3.16}$$

This objective function is established empirically. From the experimental results, we find that this objective function can achieve low clock skew effectively.

We can find the maximum of $f_{u,v}$ subject to changes in the load capacitance values and a set of linear equality constraints as described below.

The sink load capacitances have upper and lower bounds according to the distribution of the capacitance values. For each sink node S_i

$$l_i < C_i < u_i \tag{3.17}$$

We also have a set of linear equality constraints to describe the delay equalities due to the links. Let $S_{left(k)}$ and $S_{right(k)}$ be the two endpoints of link k. W.l.o.g, let $S_{left(k)}$ be the node being split and $S_{left(k)}$ becomes $S'_{left(k)}$ and $S_{mid(k)}$ after splitting. Assume that the delays at $S'_{left(k)}$ and $S_{right(k)}$ are t_{k1} and t_{k2} respectively, and the delay at $S_{mid(k)}$ is t_{k3}. $D^k{}_{link}$ is the delay on the link connecting $S_{mid(k)}$ and $S_{right(k)}$. Assume that the link resistance is ${R^k}_{link}$, the equality constraints due to the k^{th} link is given by:

$$t_{k1} = t_{k3} = t_{k2} + {D^k}_{link} \tag{3.18}$$

$${D^k}_{link} = {R^k}_{link} c_k \tag{3.19}$$

To summarize, the optimization problem to maximize the objective function $f_{u,v}$ between node S_u and S_v is:

$$\text{Maximize:} \qquad f_{u,v} = \frac{|t_u - t_v|}{r_{u,v}} \tag{3.20}$$

Subject to:

$$t_u = f_u(C_1, C_2, \ldots, C_n, c_1, c_2, \ldots, c_m) \tag{3.21}$$

$$t_v = f_v(C_1, C_2, \ldots, C_n, c_1, c_2, \ldots, c_m) \tag{3.22}$$

$$l_i < C_i < u_i \qquad i = 1, \ldots, n \tag{3.23}$$

$$t_{k1} = t_{k2} + {R^k}_{link} c_k \qquad k = 1, \ldots, m \tag{3.24}$$

where C_i for $i = 1, \ldots, n$ and c_j for $j = 1, \ldots, m$ are variables in the LP. The absolute value in equation (3.20) is handled in the implementation. For each pair (i, j), we do optimizations twice. In one of them, we maximize $(t_u - t_v)$ to obtain the positive maximum. And in the other one, we minimize $(t_u - t_v)$ to obtain the negative minimum. In our implementation, instead of calling a linear solver, we will first use Gaussian elimination to solve the k equality constraints (3.24), so the k variables c_i can be expressed by linear combinations of $C_1, C_2, \ldots, C_n$. Equation (3.21) and (3.22) become

$$t_u = {f_u}'(C_1, C_2, \ldots, C_n) \tag{3.25}$$

$$t_v = {f_v}'(C_1, C_2, \ldots, C_n) \tag{3.26}$$

Then the maximum objective value subject to changes in the sink load capacitances can be computed directly from the upper and lower capacitance bounds.

The maximum value of $f_{u,v}$ (called $p_{u,v}$) for a pair of node S_u and S_v can be obtained by solving the above linear program. Finally, we can find a node pair with the largest $p_{u,v}$, which is denoted as p_{max}. A cross link is added between this node pair to reduce the clock skew while constraining the wire length of the link. By applying the technique recursively, we can add a user-defined number of cross links to construct a link based clock network.

3.2.4 Reducing the Number of Optimizations

Although the optimization problem for a pair of nodes can be solved quickly as formulated above, we need to run it for all pairs of nodes to get the node pair with p_{max}. The running time will increase quickly when the number of nodes increases to thousands. Therefore, we devised a method to speed up the process when the number of sinks is large. In order to reduce the running time, instead of trying all pairs of sinks, we will choose some node pairs which are more likely to have the p_{max}. Actually there are some pairs that we do not need to consider, e.g., nodes that are topologically close to each other. This is because these node pairs share a long mutual sub-path from the source and the delay difference between them will not be big.

Starting from the sink node level, we will travel up to find a set of internal nodes I such that the set S of subtrees rooted at these internal nodes cover all the sink nodes. The detailed procedure to find the internal nodes is shown in Fig.3.5. Compared with the total number of sinks, the number of these internal nodes will be smaller. Since the sink nodes within a subtree in S will likely to have similar delays, we do not consider the skews among these internal nodes. We will pick one node from each subtree in S in order to find the p_{max} in the clock network, as illustrated in Fig.3.6. In our implementation, we consider the number of sinks in each subtree in S and limit them to be

```
Algorithm: Find Subtrees
Input: A clock network
Output: Internal node set $I$ that reduce the number of optimizations
1. $I = \Phi$
2. For each sink node $S_i$
3.   If $S_i$ is contained in a subtree rooted at a node in $I$
4.       Continue
5.   Else
6.       $r \leftarrow 1$, $I_i \leftarrow S_i$
7.       While $r <$ lower bound in the sink number of a subtree
8.           $last\_I_i \leftarrow I_i$, $I_i \leftarrow$ parent of $I_i$
9.           $r \leftarrow$ number of sinks contained in the subtree rooted at $I_i$
10.      End
11.      If $r >$ upper bound in the sink number of a subtree
12.          $I_i \leftarrow last\_I_i$
13. Add $I_i$ to $I$
```

Figure 3.5: Find a Set of Internal Nodes I

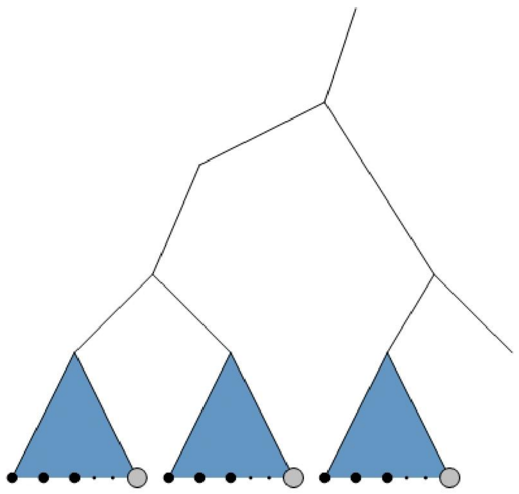


Figure 3.6: Reducing the number of nodes

less than or equal to 16. In this way, even benchmarks with a large number of sinks can be solved in a reasonable amount of time.

3.2.5 Experimental Results

Benchmark and Experimental Setup

Our algorithm to select node pairs for non-tree clock network construction with cross links is implemented in C. The experiments are performed on a 3.2 GHz Intel CPU Linux machine with 4GB memory. The key difference between our work and the existing works is that most of existing works did not consider non-tree delay when inserting links, which inevitably limits the effectiveness of the link insertion step.

To verify the effectiveness of our method, we will compare our link insertion scheme with the algorithm proposed in [1]. We implemented the minimum weight matching-based link insertion method which is the best method in [1]. We start with the same zero skew clock tree which is obtained from the bounded skew tree method [2]. Both the benchmarks and the bounded skew tree code are downloaded from the GSRC bookshelf [21].

In our implementation and simulations, the per unit wire resistance is 10^{-4} Ohm/nm and the per unit wire capacitance is 2×10^{-4} fF/nm. Ngspice is used to simulate our clock distribution network. Process variation is accounted in the simulations for VDD variation and sink capacitance variation. We allow 15% variation in the VDD and the sink capacitances and all the variations follow a normal distribution. For each clock network, 500 spice simulations are performed to obtain the worst case skew (WCS).

Table 3.1: Benchmark information

BM	#Sinks	WCS (ps)	Cap (fF)
r1	267	0.364	264.1
r2	598	0.802	523.7
r3	862	1.250	678.2
r4	1903	2.504	1357.8
r5	3101	2.612	2005.5
s1423	74	0.048	21.30
s5378	179	0.061	35.10
s15850	597	0.136	89.65

Results and Discussion

All the benchmark information are shown in Table 3.1 and the experimental results of our work and the method in [1] are shown in Table 3.2. The benchmark sizes, worst case skews (WCS), and the wire capacitances of the clock trees are given. We list the number of links inserted, WCS results, link capacitance results of the Link-M method and our work. The Link-M method refers to the minimum weight matching based link insertion method in [1]. The numbers inside the brackets are the ratios of methd in [1] and our work. The CPU time is listed in seconds in the table.

From Table 3.2, we observe that our method always outperforms the Link-M method in terms of clock skew reduction. Besides, for a same number of links, Link-M method costs much more link capacitance comparing with our algorithm. On average, we achieve 28% clock skew reduction with only 40% link resources. The reason is that the Link-M method does not consider non-tree delay when inserting links, which may result in ineffective cross link insertion into the clock tree. By considering the balance between clock skew and link resistance, we can achieve more clock skew reduction with less link capacitance resource. The CPU time for the Link-M method is ignorable so it is not listed in the table. We use the technique discussed in Section 3.2.4 to control the running time and we can see that our running time is acceptable even for the largest benchmark.

□ **End of chapter.**

Table 3.2: Worst case skew and wire cap results with 15% variations in vdd and sink capacitances

BM	#Sinks	Links	Link-M WCS (ps)	Our WCS (ps)	Link-M Cap (fF)	Our Cap (fF)	CPU (s)
r1	267	4	0.314 (1.33)	0.236 (1)	9.7 (3.24)	2.9 (1)	0.81
		16	0.162 (1.41)	0.115 (1)	16.3 (1.23)	13.2 (1)	3.21
		22	0.122 (1.20)	0.102 (1)	60.0 (3.14)	19.1 (1)	4.52
r2	598	4	0.772 (1.68)	0.460 (1)	6.9 (4.93)	1.4 (1)	9.23
		20	0.286 (1.39)	0.206 (1)	21.0 (1.60)	13.1 (1)	46.47
		40	0.192 (1.42)	0.135 (1)	40.7 (1.46)	27.9 (1)	94.09
r3	862	6	0.470 (1.09)	0.433 (1)	10.2 (1.96)	5.2 (1)	2.74
		24	0.305 (1.08)	0.282 (1)	25.9 (1.07)	24.3 (1)	11.73
		48	0.229 (1.36)	0.168 (1)	101.6 (1.91)	53.2 (1)	27.21
r4	1903	8	1.565 (1.55)	1.008 (1)	41.6 (8.67)	4.8 (1)	9.21
		40	0.448 (1.21)	0.370 (1)	57.3 (1.23)	46.5 (1)	64.34
		68	0.336 (1.03)	0.327 (1)	125.3 (1.42)	88.3 (1)	140.3
r5	3101	8	1.451 (1.14)	1.276 (1)	43.2 (7.20)	6.0 (1)	36.17
		40	0.740 (1.14)	0.648 (1)	76.2 (1.67)	45.7 (1)	229.8
		72	0.589 (1.16)	0.506 (1)	95.9 (1.08)	89.0 (1)	510.5
s1423	74	4	0.042 (1.83)	0.023 (1)	1.33 (1.17)	1.14 (1)	0.03
		8	0.036 (2.57)	0.014 (1)	2.77 (1.07)	2.60 (1)	0.03
s5378	179	4	0.063 (1.75)	0.036 (1)	1.60 (2.86)	0.56 (1)	0.29
		12	0.025 (1.09)	0.023 (1)	5.94 (2.57)	2.31 (1)	0.89
s15850	597	4	0.111 (1.34)	0.083 (1)	0.56 (1.27)	0.44 (1)	10.62
		22	0.053 (1.39)	0.038 (1)	3.88 (1.56)	2.48 (1)	59.82
AVG			0.396 (1.39)	0.309	35.6 (2.49)	21.4	60.10

Chapter 4

Buffered Clock Network with Cross Links

4.1 Link Insertion in Buffered Clock Network

The cross link insertion algorithm proposed in [1] is applied to unbuffered clock tree. In the real world, buffer plays a key role in the design of a clock system. Buffers are used in a clock network to reduce signal delay and to preserve clock waveform [31]. Several works [23] [24] [32] address the problem of buffer insertion in a clock network. Cross link insertion in a buffered clock network to achieve target skew is more challenging in comparison with the unbuffered case. It is claimed in [10] that the effect of cross link on skew variability still holds for a buffered clock tree. We also want to extend our cross link insertion to buffered clock tree. The major difference between buffered and unbuffered clock tree for our linear program approach is the formulation of delay. To handle a clock network with buffers, we must consider the buffering effect when formulating the Elmore delay in equation (3.14) and

(3.18):

$$t_u = f_u(C_1, C_2, \ldots, C_n, c_1, c_2, \ldots, c_m)$$
$$t_{k1} = t_{k3} = t_{k2} + {D^k}_{link}$$

where $C_1, C_2, \ldots, C_n$ are the sink capacitances and $c_1, c_2, \ldots, c_m$ are m variables describing the load redistribution from one endpoint of a link to the other. t_{k1} and t_{k2} are the delays at $S'_{left(k)}$ and $S_{right(k)}$ respectively. t_{k3} is the delay at $S_{mid(k)}$. ${D^k}_{link}$ is the delay on the link connecting $S_{mid(k)}$ and $S_{right(k)}$.

4.1.1 Delay Calculation in Buffered Clock Network

Tsay [16] proposes a method computing Elmore delay in a clock tree with linear time complexity. Let c_i be the capacitance of node i and r_i be the resistance of edge i in a clock tree. Edge i is defined as the edge between node i and its parent. Denote by $IS(i)$ the set of all immediate successors of node i, the subtree capacitance C_i of T_i is

$$C_i = c_i + \sum_{k \in IS(i)} C_k$$

The subtree capacitance can be computed in a recursive bottom up fashion. To calculate delay, assuming that node i is a predecessor of node j and $N(i, j)$ is a collection of nodes on the path between node i and node j, the delay time t_{ij} between node i and node j is

$$t_{ij} = \sum_{n \in N(i,j)} r_n C_n$$

The delay to each node can be calculated from the delay to its parent, the edge resistance and the subtree capacitance. To account for a buffered RC tree, the subtree capacitance is modified as

$$C_i = \begin{cases} c_i & \text{if node } i \text{ is associated with a buffer} \\ c_i + \sum_{k \in IS(i)} C_k & \text{otherwise} \end{cases}$$

Delay computation between a node i and its successor j is extended as

$$t_{ij} = \sum_{n \in N(i,j)} r_n C_n + d_n$$

where d_n is the buffer internal delay. The calculation is done in the same way with an appropriate buffer modeling and remains linear time complexity.

4.1.2 Linear Program Formulation for Buffered Clock Network

Linear program for inserting cross links into a buffered clock network is similar to that discussed in Section 3.2.3:

$$\text{Maximize:} \qquad f_{u,v} = \frac{|t_u - t_v|}{r_{u,v}}$$

Subject to:

$$\begin{gathered} t_u = f_u(C_1, C_2, \ldots, C_n, c_1, c_2, \ldots, c_m) \\ t_v = f_v(C_1, C_2, \ldots, C_n, c_1, c_2, \ldots, c_m) \\ l_i < C_i < u_i \qquad i = 1, \ldots, n \\ t_{k1} = t_{k2} + R^k{}_{link} c_k \qquad k = 1, \ldots, m \end{gathered}$$

where the objective function is a balance between the worst case skew and the link resistance $r_{u,v}$. $l_i < C_i < u_i$ describe upper and lower bounds of the sink load capacitances. $t_{k1} = t_{k2} + R^k{}_{link} c_k$ is a set of linear equality constraints to describe the delay equalities due to the links.

When we formulate the delays t_u, t_v, t_{k1} and t_{k2} in the linear program, we need to consider the buffering effect. The buffer model used in our work follows the ISPD contest benchmark [15], which has three parameters: an input load capacitance, an output parasitic capacitance and an output resistance. As discussed in Section 3.2, for each sink node pair considered for link insertion, we find the maximum possible skew under capacitance variation. A link will be inserted to the node pair with the maximum possible skew. The linear program is then updated to find the next node pair for link insertion. This process will continue until a user defined number of links are added into the clock network.

4.2 Experimental Results and Comparison

We compare our work with a recent approach in [4] which inserts cross links into the internal nodes of a tree while constructing the clock network. The experiments are performed on the ISPD 2010 contest benchmark. The per unit wire resistance is 10^{-4} Ohm/nm and the per unit wire capacitance is 2×10^{-4} fF/nm. Process variation is accounted in the simulations with 15% variation in the VDD and 15% variation in sink capacitances. All the variations follow a normal distribution. For comparison with [4], we use the Local Clock Skew (LCS) as a metric to measure the skew variability instead of using the Worst Case Skew (WCS). Local clock skew is the clock skew between any two sinks within a certain distance from each other, which is defined in the ISPD 2010 High Performance Clock Network Synthesis Contest [15]. For each clock network, 500 ngspice simulations are performed to obtain the worst Local Clock Skew (LCS).

Table 4.1 lists the results that compares our work with the method in [4]. We list the benchmark size, the LCS results, the capacitance usage and the

Table 4.1: Comparison of our method with the work in [4]

Bench -mark	#Sinks	Method	LCS (ps)	Ratio of Link Cap	CPU (s)
01	1107	[4]	7.88	1.007	1092
		Our work	10.67	1	52.4
02	2249	[4]	8.32	1.223	4314
		Our work	8.17	1	424
03	1200	[4]	6.34	1.073	383
		Our work	5.89	1	10.6
04	1845	[4]	7.42	1.001	934
		Our work	7.33	1	33.8
05	1016	[4]	5.90	1.199	278
		Our work	5.87	1	5.36
06	981	[4]	6.78	1.003	285
		Our work	8.98	1	6.69
07	1915	[4]	6.77	1.228	818
		Our work	6.05	1	49.8
08	1134	[4]	6.42	1.225	327
		Our work	8.59	1	8.34

running time. Note that the CPU time of our work in the table is for the link insertion phase only, while the tree construction time is not included.

We obtain the clock network results from the authors of [4]. Firstly, the links added between the internal nodes using their algorithm are removed. Then our method is applied to generate cross links into the clock network. In general, our work uses less link resources to achieve similar LCS results. Please note that the two results are not directly comparable, because in [4], the links are inserted while the trees are being constructed, so their tree construction performs in such a way to favor their link insertion step. In our case, we take their trees and insert the links in a post processing way, so the results are hard to be compared. However, we still want to display the comparisons to show that our method can also handle clock trees with buffers and can perform actually quite well comparing with [4] in which the link insertion and the tree construction are performed simultaneously.

4.3 Possible Extensions

4.3.1 Link Insertion at Internal Nodes

The algorithm in [4] inserts cross links between the internal nodes of a clock tree instead of sink nodes. Their link insertion is integrated into the clock tree construction process. In addition to reducing the skew variability, their approach also try to minimize the total cross link length. Our method proposed in Section 4.1 inserts cross links between sink node pairs in a buffered clock network. We have explored whether our method can be extended to insert cross links between internal nodes of a tree as well.

The linear program formulated in Section 3.2 computes the maximum possible skew for each sink node pair. Then a link will be inserted to the node pair with the largest such value. Linear program is then updated for the next link insertion. This process continues until a user defined number of links are inserted into the clock network.

However, our method cannot be easily extended to internal cross link insertion. This is because the load redistribution effect of an inserted cross link can not be easily seen from the delay of the sink nodes when the link is inserted between two internal nodes. Consider the example in Fig.4.1, when we select a sink node pair (S_u, S_w) after solving the linear program, a corresponding internal cross link is added between two zero skew nodes u and w on the buffered wire segments below the Nearest Common Ancestor node. Inserting between two zero skew nodes helps to aviod effecting the nominal skew of the clock network. When we formulate the linear program under sink capacitance

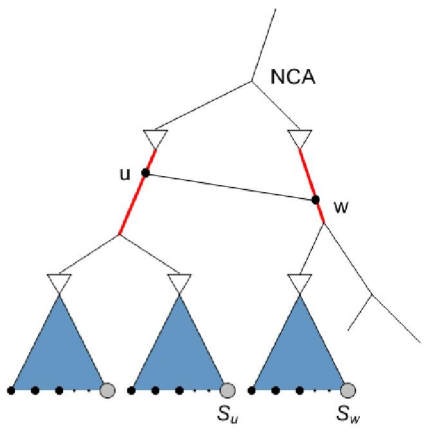


Figure 4.1: Cross Link Between Internal Nodes in a Clock Network

variation, the Elmore delay of internal nodes above the buffer does not change as it depends on the buffer rather than the sink capacitances. Therefore, the skew variation on sink nodes cannot be addressed by the internal cross link and there is no load redistribution effect through the link.

We can model other variations such as wire width variation and buffer delay variation to make internal link insertion possible. For example, if we model wire width variation in our problem formulation, the node delay will become a function of both sink capacitance variation and wire width variation. The node delay above the buffers after internal link insertion will change accroding to the variations. The internal cross link can be used to redistribute load and therefore reduce skew variability.

4.3.2 Modeling Clock Buffer Delay Variation

In our problem formulation, we model sink capacitance as source of variation. On the other hand, clock buffer delay variation is very critical to clock skew variability. We can also model clock buffer delay variation in our linear program. As discussed in section 4.1.1, the delay between a node i and its successor j in a buffered clock network is

$$t_{ij} = \sum_{n \in N(i,j)} r_n C_n + d_n$$

where r_n is the edge resistance, C_n is the subtree capacitance and d_n is the buffer internal delay. If we model buffer internal delay d_n as randam variation, assuming that there are l buffer internal delays, we can still write the delay as a linear function of the variations as follows

$$t_u = f_u(C_1, C_2, \ldots, C_n, c_1, c_2, \ldots, c_m, d_1, d_2, \ldots, d_l)$$

Thus, our problem formulation can be extended to handle the clock buffer delay variation as well.

□ **End of chapter.**

Chapter 5

Conclusion

Reducing clock skew caused by variation is now one of the most important problems in clock network synthesis. Link based clock network is considered to be a promising way to handle the skew variation problem. In this thesis, we explore where links should be inserted into a clock network to achieve the most effective skew reduction. Our contributions are summariez as follows:

Delay Calculation in Non-tree Clock Network

In our approach, a non-tree clock network is decomposed into a tree with some way of load redistribution. We are able to formulate the signal delay and clock skew in the network. Based on this delay information, our new method selects node pairs analytically for cross link insertion. Compared with existing works on this cross link problem, this analytical delay calculation in node pair selection results in more effective link insertion.

Solving Problem with an Efficient Linear Program

We formulate the problem as a linear program, with an objective function considering tradeoff between clock skew reduction and link length. For each sink node pair considered, we formulate a LP to find the maximum possible skew value under capacitance variation. A link will be added to the node pair with the largest such value. The linear program is incrementally updated and solved until a user defined number of links are inserted into the clock network.

Effectiveness of Our Approach

We devise a way to reduce the number of optimizations and use a Gaussian elimination based method to speed up solving the linear program. Therefore, even the largest benchmark can be solved in just a few miniutes. Experiments on the two sets of benchmarks verified the effectiveness of our approach. We achieve 28% clock skew reduction with only 40% link resources. Our method can be applied to insert links very effectively while reducing the total wire length.

□ **End of chapter.**

Bibliography

[1] A. Rajaram, J. Hu, and R. Mahapatra. Reducing clock skew variability via crosslinks. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, 2006, 1176-1182.

[2] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao. Bounded-skew clock and steiner routing under Elmore delay. In *Proceedings of the 1995 IEEE/ACM International conference on Computer-aided design*, 1995, 66-71.

[3] P. Chan and K. Karplus. Computing signal delay in general RC networks by tree/link partitioning. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 8, 1990, 898-902.

[4] T. Mittal and C.-K. Koh. Cross link insertion for improving tolerance to variations in clock network synthesis. In *Proceedings of the 2011 international symposium on Physical design*, 2011, 29-36.

[5] P. Restle, T. McNamara, D. Webber, P. Camporese, K. Eng, K. Jenkins, D. Allen, M. Rohn, M. Quaranta, D. Boerstler, et al.. A clock distribution network for microprocessors. *IEEE Journal of Solid-State Circuits*,vol. 36, no. 5, 2001, 792-799.

[6] G. Venkataraman, Z. Feng, J. Hu, and P. Li, Combinatorial algorithms for fast clock mesh optimization. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*, 2006, 563-567.

[7] H. Chen, C. Yeh, G. Wilke, S. Reddy, H. Nguyen, W. Walker and R. Murgai. A sliding window scheme for accurate clock mesh analysis. In *Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, 2005, 939-946.

[8] A. Rajaram and D. Z. Pan. MeshWorks: An efficient framework for planning, synthesis and optimization of clock mesh networks. In *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, 2008, 250-257.

[9] W. -C. D. Lam, J. Jain, C. -K. Koh, V. Balakrishnan, and Yiran Chen. Statistical based link insertion for robust clock network design. In *Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, 2005, 588-891.

[10] G. Venkataraman, N. Jayakumar, J. Hu, P. Li, S. Khatri, A. Rajaram, P. McGuinness, and C. Albert. Practical techniques for minimizing skew and its variation in buffered clock networks. In *Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, 2005, 592-596.

[11] Anand Rajaram and David Z. Pan. Variation tolerant buffered clock network synthesis with cross links. In *Proceedings of the 2006 international symposium on Physical design*, 2006, 157-164.

[12] Z. Li, Y. Zhou, and W. Shi. Wire sizing for non-tree topology. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 5, 2007, 872-880.

[13] T. M. Lin and C. A. Mead. Signal delay in general RC networks. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 3, no. 4, 1984, 331-349.

[14] P. K. Chan and M. D. F. Schlag. Bounds on signal delay in RC mesh networks. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 6, 1989, 581-589.

[15] http://archive.sigda.org/ispd/contests/10/ispd10cns.html.

[16] R. Tsay. Exact zero skew. In *Proceedings of IEEE International Conference on Computer-Aided Design*, 1991, 336-339.

[17] K.D. Boese and A.B. Kahng. Zero skew clock routing trees with minimum wire length. In *Proc. IEEE int. conference ASIC*, 1992, 1.1.1-1.1.5.

[18] T. H. Chao, Y. C. Hsu, and J. M. Ho. Zero skew clock net routing. *ACM/IEEE Design Automation Conference*, 1992, 518-523.

[19] Masato Edahiro. Minimum skew and minimum path length routing in vlsi layout design. *NEC, Res Devel*, 1991, 32 4:569-575.

[20] P. Restle, K. Jenkins, A. Deutsch, P. Cook. Measurement and Modeling of On-chip Transmission Line Effects in a 400MHz Microprocessor. *IEEE Journal of Solid-State Circuits*, 1998, 662-665.

[21] http://vlsicad.eecs.umich.edu/BK/.

[22] J. Rabaey, A. Chandrakasan, and B. Nikolic. Digital Integrated Circuits: A Design Perspective. Prentice-Hall, 2nd Edition, 2002.

[23] Y. P. Chen and D.F. Wong. An Algorithm for Zero-Skew Clock Tree Routing with Buffer Insertion. In *Proceedings of the 1996 European conference on Design and Test*, 1996, 230-236.

[24] B. Wu and N. A. Sherwani. Effective Buffer Insertion of Clock Trees for High-speed Vlsi Circuits. *Microelectronics*, 1992, 291-300.

[25] Linfu Xiao, Zigang Xiao, Zaichen Qian, Yan Jiang,Tao Huang, Haitong Tian, and Evangeline F.Y. Young. Local Clock Skew Minimization Using Blockage-Aware Mixed Tree-Mesh Clock Network. In *Proc. of the ICCAD*, 2010.

[26] M. A. B. Jackson, A. Sirinivasan, and E.S. Kuh. Clock Routing for High-performance ICs. In *Proceedings of 27th ACM/IEEE Design Automation Conference*, 1990, 573-579.

[27] A. Kahng, J. Cong, and G. Robins. Matching based models for high performance clock routing. *IEEE Transactions on CAD of Integrated Circuits and Systems*, 1993, 1157-1169.

[28] Fuqiang Qian, Haitong Tian, Evangeline F. Y. Young. Crosslink insertion for variation-driven clock network construction. *ACM Great Lakes Symposium on VLSI 2012*, 2012 321-326.

[29] K. Jain. A Factor. Approximation Algorithm for the Generalized Steiner Network Problem. In *IEEE Symposium on Foundations of Computer Science*, 1998, 448-457.

[30] H. Kerivin and A. R. Mahjoub. Design of Survivable Networks: A survey. In *Networks*, 2005, 1-21.

[31] Naveed A. Sherwani. Algorithms for VLSI Physical Design Automation. Kluwer Academic Publishers, 2nd Edition, 1995.

[32] R. Chaturvedi and J. Hu. Buffered clock tree for high quality IC designs. In *Proc. IEEE International Symposium on Quality Electronic Design*, 2004, 381-386.

[33] J. Cong, A. B. Kahng, C.-K. Koh, and C.-W. A. Tsao. Bounded-skew clock and Steiner routing. *ACM Transactions on Design Automation of Electronic Systems*, 1998, 341-388.