Minimizing Inter-Clock Coupling Jitter

Ming-Fu Hsiao, *Malgorzata Marek-Sadowska, Sao-Jie Chen
Department of Electrical Engineering, National Taiwan University, Taiwan, R.O.C.
*Department of Electrical and Computer Engineering,
University of California, Santa Barbara, CA 93106, USA

Abstract

Crosstalk noise is a crucial factor affecting chip performance in deep submicron technologies. Among all possible crosstalk noise sources, clock is the most common aggressor as well as victim. Crosstalk on clock nets can increase clock jitter, which may degrade significantly the system performance. Besides, in modern chip designs, there is usually more than one clock net, and sometimes even tens of them. It is therefore imperative to design clock topologies to prevent possible crosstalk among them. In this paper, we address the inter-clock crosstalk. We propose algorithms to design clock topology and to perform routing minimizing the effective crosstalk. Our experimental results show a significant reduction of clock jitter compared to the conventional clock tree synthesis which does not take into account the inter-clock crosstalk effects.

1. Introduction

With the recent advances in VLSI technology, the device sizes have shrunk below 0.1um. The shrinking geometries have brought two new major concerns for signal integrity. One is the power and ground levels fluctuations caused by simultaneous switching circuits. The other problem is the increasing aspect ratio of wires and the decreasing of interconnect spacing which have made coupling capacitance larger than self-capacitance. The ratio of coupling capacitance is reported to be as high as 70%-80% of the total capacitance. For high speed circuits, it has become very important to decrease the coupling effect during the layout phase.

Clock synthesis has been an important step in chip design for more than ten years. Synchronous designs have been the most popular as they are robust and easy to migrate. High integration of chip functions has increased the number of clocks from one to as many as ten. To balance the trade-off between performance and cost, chips are often designed with hierarchies of clocks. For example, consider the on-chip memory. Registers offer the highest frequency memory and are synchronized with CPU. But registers can provide only a small amount of the memory requirement. Cache memory is often used as the second choice, other low speed memory such as DRAM is the third choice, and so on. Different types of memory use different synchronous circuits, thus increasing the number of clocks. Others, like interface circuits or different functional units may also run at different frequencies, thus increasing the number of clocks on a single chip. Clocks distributed throughout the chip toggle in every cycle with the highest frequencies; therefore improper layout could easily introduce too much crosstalk noise and cause chip failure[5]. Coupling can be accumulated over large portions of a clock tree and hence may be very significant. Fig 1. shows a typical clock signal propagating through a series of buffers. In each stage, crosstalk increases the timing uncertainty, or jitter. The clock jitter increases and accumulates throughout the tree. Thus at the last stages, it can be very significant. Our experiments show that for a ten stage clock signal, the jitter can grow from 0 at the input to 293ps (Table 3) at the leafs. Failure cases have been reported due to the cross-coupling of clock signals. Since the clock network is always synthesized after placement and before signal routing, it makes sense to minimize the mutual coupling during the clock synthesis phase. The smaller the effective coupling in this phase, the better clock quality will be.

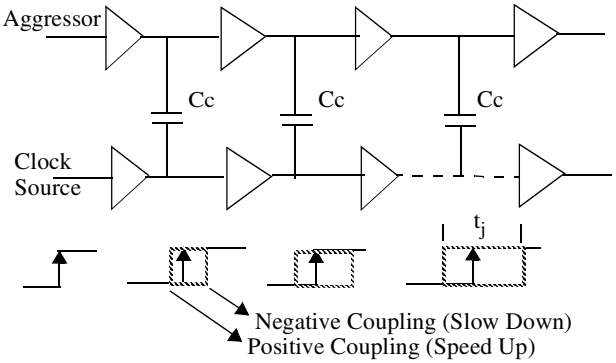


Fig. 1: Crosstalk effect on a multi-stage clock signal. The timing uncertainty is magnified at every stage. t_j is the clock timing uncertainty(jitter).

Shielding is traditionally an efficient way to prevent clock from crosstalk effect. Complete shielding of clock net can completely remove the clock coupling. However complete shielding of clock net has two major disadvantages. First of all, complete shielding will triple the routing resource since two additional parallel metal wires have to be placed aside a clock wire. Secondly, the shielding wires have to be tied to ground and hence increase the lump capacitance and clock delay. In this paper, we demonstrate that proper clock routing can significantly reduce the need of clock shielding and hence save routing resource.

In this paper, we propose the crosstalk minimization strategy for the multiple clock routing problem. As far as we know, ours is the first attempt to attack this problem. The problem formulation is given in Section 2, followed by the clock topology and routing algorithms in Sections 3, 4 and 5. Experimental results are given in Section 6, Section 7 summarizes the key points of the paper.

2. Problem Formulation

There are several factors which determine crosstalk noise. Some, such as the victim's driving strength and input capacitance, are related to the circuit design. Others, such as the slew or frequency of aggressor's signals, are related to the waveform of input stimulus. Analytical methods have been established to characterize the behavior of crosstalk noise[2][7]. The only factor affecting crosstalk that is related to layout is the coupling capacitance, which is a function of spacing and coupling length. Fig. 2 shows the relationship between a unit coupling capacitance and the spacing for a typical 0.13um CMOS process. One can observe that the unit cou-

The second author acknowledges partial NSF support through grant CCR0098069.

pling capacitance drops dramatically with increased spacing. When the spacing is more than 10um, then the effect of coupling capacitance becomes negligible. Here we define the effective coupling as follows.

Definition 1: Effective coupling length is the length of two signals running in parallel whose spacing is smaller than S_{max}. We define S_{max} as the threshold spacing value below which the effect of the crosstalk is not negligible.

The distance at which this S_{max} spacing occurs is the effective coupling length. The accumulated effective coupling length can be used to measure the quality of layout. The longer the accumulated coupling length, the more affected by crosstalk the signal will be. So a good router should prevent coupling capacitance from accumulating. We define the crosstalk minimization for multiple clock routing problem as follows.

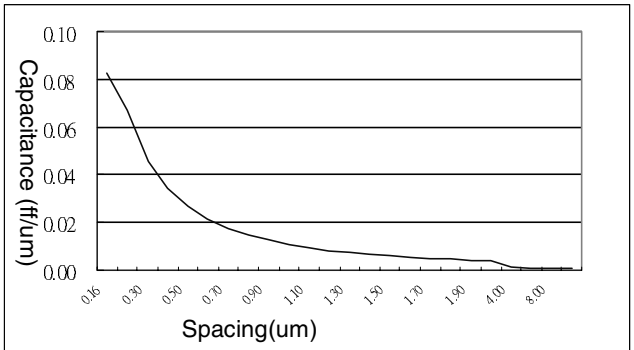


Fig. 2: Relationship between spacing and unit coupling capacitance for a typical 0.13um CMOS process

Problem 1: **Crosstalk Minimization for Multiple Clock Tree Routing:** Given a set of clock domains, T_1, T_2,..., T_n, with sink nodes for each, the problem is to construct the clock trees such that: (1) for each tree the total effective coupling length is minimized, (2) the zero skew in each clock tree is maintained for the given delay model, (3) the total wire length is minimized.

Our formulation adds one more requirement on clock routing, as the second and third requirements are typical.

We propose a two-phase approach to generate the clock topology followed by a minimal crosstalk two-pin net detailed router. In the bottom-up phase, we decide the topologies and merging segments simultaneously, level by level, for all clocks. As a cost function we use the routing area overlap among the trees. At each level we formulate and solve a minimum-cost perfect matching (MCPM) problem. In the second phase, after the topologies have been decided, we assign the merging points, level by level, on the merging segments to minimize the effective coupling length. At each level, the problem is formulated as a placement problem which is solved by a simulated annealing approach.

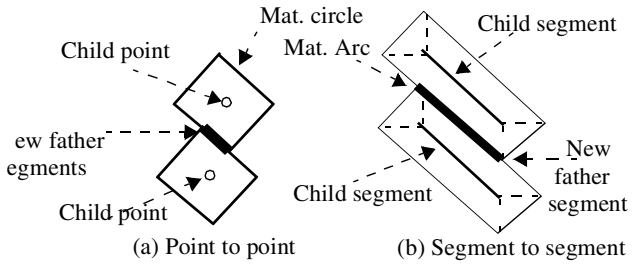


Fig. 3: Merging segments for two points and two segments

3. Bottom-Up Phase

A collection of points at the same Manhattan distance from a given point forms a Manhattan circle. The shape of the Manhattan circle is a square with a 45-degree slope. Given two points, the zero skew merging points are at the intersection of two Manhattan circles (Fig 3(a)). A radius of the Manhattan circle depends on the delay model. Its length is selected to maintain a zero skew. This length is maintained and will be used in the top-down phase. The intersection of Manhattan circles is a segment with slope of 1 or -1, called the Manhattan arc. It has been proven in [1][6] that for two given merging segments (Manhattan arcs), the zero skew merging points also form a Manhattan arc. An example is given in Fig 3(b).

The algorithm starts from the bottom-up phase and constructs the tree topology level by level. At each level, each node is matched to another node such that the overall cost function is minimized.

3.1 Neighbor Graph

One of the major goals for a clock-routing algorithm is to generate a topology of minimum wire length. We assume that during the tree construction process, each node can be matched to only one of its nearest neighbors. Matching remote nodes would likely increase the tree lengths. A constant N denotes the number of the nearest neighbors we consider for each node.

Definition 2: Neighbor Graph: Each merging segment (or a point for sink nodes) corresponds to a node in the graph. An edge E_{ij} exists between two vertices V_i and V_j, if V_j is among the N nearest neighbors of V_i or V_i is one of the N nearest neighbors of V_j. The cost of an edge is defined in subsection 3.3.

Definition 3: Distance Between Segments: The distance between two segments is the minimum distance between two points on these segments.

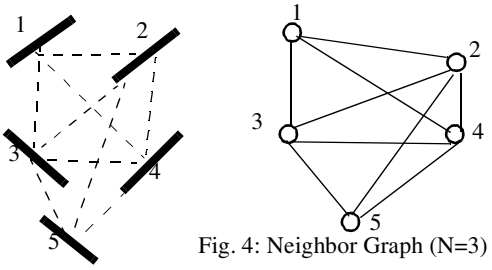


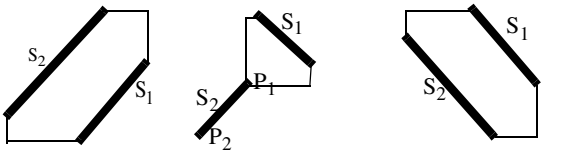
Fig. 4: Neighbor Graph (N=3)

An example is given in Fig. 4; there, N=3. Some of the nodes might have more than 3 edges. For instance, node 2 is one of the three nearest neighbors of node 5, but node 5 is not one of the three nearest neighbors of node 2. Node 2 has four edges. So does node 4.

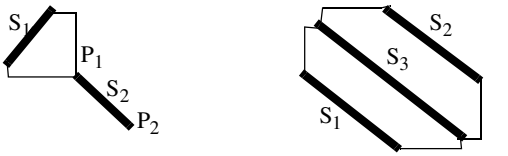
3.2 Routing Region

A routing region defines the potential routing area of two parent-child segments. Since the slope of merging segments can be only 1 or -1, there are just four types of routing regions that correspond to merging (1, 1), (-1, -1), (1, -1), and (-1, 1) slope segments. These four types are illustrated in Fig 5. Note that in (b) and (d) one of the end points (P_2) of the segment doesn't contribute to the routing region because we assume S_1 is the father of S_2. So all the points

on S_1 have equal Manhattan distance to P_1, and the distance is smaller than the distance to P_2. Thus only P_1 will be considered as the shortest merging point.



(a) Type 1, (1, 1) (b) Type 2, (-1, 1) (c) Type 3, (-1, -1)



(d) Type 4, (1, -1) (e) Routing region of adjacent node S_1, S_2

Fig. 5: Examples of routing regions

3.3 Edge Cost

An edge E in the neighbor graph implies a possible parent-merging segment for the two neighbors. For example, in Fig 5(e) if there is an edge between S_1 and S_2, a new merging segment S_3 is introduced corresponding to (S_1, S_2), which forms two routing areas, (S_1,S_3) and (S_2,S_3). The area is denoted by A(E). Supposing there are M trees, the routing area of the ith edge E_{ij} in graph G_j may overlap with the routing area of the kth edge E_{kl} in graph G_l. The intersection area of these two edges is denoted by $RC(E_{ij},E_{kl})$. And $RC(E_{ij},E_{kl})=0$ if j=l, because we are concerned with the overlapping of different clocks. We define the routing collision ratio for an edge as follows.

Definition 4: Routing Collision Ratio: Suppose there are M neighbor graphs G_1, G_2,..., G_M corresponding to M clocks. For the edge E_{ij} in the graph G_j, the routing collision ratio of E_{ij} is defined as $ERC = \frac{1}{A(E_{ij})}\sum_k^{j\neq l} RC(E_{ij},E_{kl})$. It is the ratio of the overlapping area to its own area. We define the cost of an edge as $C(E) = \alpha \times K \times ERC + \beta \times L$ where α is the weight of the routing collision ratio, β is the weight of the wire length, L is the distance between its two descendant children, and K is the normalization factor for ERC and L. For conventional clock tree synthesis minimizing the total wire length, α is set to 0, and β is set to 1.

3.4 Minimum Cost Perfect Matching

At each level, M neighbor graphs are built simultaneously, and minimum-cost perfect matching (MCPM) is performed on each of them. Below is the description of the bottom-up process.

Procedure Bottom-Up Phase
while roots of all clocks not reached, do
Build neighbor graph for each clock;
Update the edge cost;
Do MCPM on each tree;
Calculate parent merging from the matching result;
Move one level up.
End while
End procedure

The output of the bottom-up phase is a binary tree of merging segments. In each intermediate node the distances to all the leaves are kept the same, based on the given delay model.

4. Top-Down Phase

After the bottom-up process, has created the topology every clock is represented as a tree of merging segments. The purpose of the top-down phase is to determine the exact locations of merging points such that the overall coupling length is minimized. Any edge in the resulting tree represents a two-pin net. Therefore, the entire tree can be expressed as a set of two-pin nets.

4.1 Cost Function for Top-Down Phase

In general, the coupling capacitance can be decided only after detailed routing. At the merging embedding stage two-pin nets are

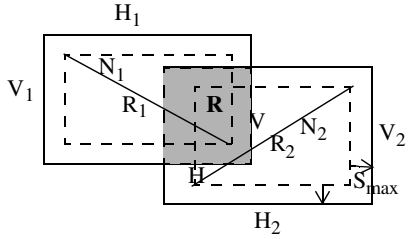


Fig. 6: Overlapping wires

sequentially routed. We need a cost function to estimate the detailed routing quality in order to select merging points. The cost function should reflect the expected value of effective coupling length after routing. We determine the expected value of effective coupling length using a probabilistic approach. We define the expected effective coupling length among two two-pin nets as follows:

Let N_1 = ($P_1(X_0,Y_0)$, $P_2(X_1,Y_1)$), and N_2 = ($P_3(X_2,Y_2)$, $P_4(X_3,Y_3)$) are two-pin nets. We extend the bounding boxes of N_1 and N_2 by S_{max} in horizontal and vertical directions. Now N_1 spans a rectangle R_1 whose width is $H_1=|X_1-X_0|+2*S_{max}$ and height is $V_1=|Y_1-Y_0|+2*S_{max}$. Similarly, N_2 spans a rectangle R_2 whose width is $H_2=|X_3-X_2|+2*Smax$ and height is $V_2=|Y_3-Y_2|+2*S_{max}$. Let R of height V and width H be an overlap rectangle between R_1 and R_2 as shown in Fig 6. The expected vertical coupling length is $V_{ecl} = V \times \frac{H}{H_1} \times \frac{H}{H_2} \times \frac{2 \times Smax}{H}$, where V is the length of the possible vertical coupling, H/H_1 is the probability that a vertical wire of R_1 falls into R, H/H_2 is the probability that a vertical wire of R_2 falls into R, and $2xS_{max}/H$ is the probability that two vertical wires are within distance S_{max}. Similarly, the expected horizontal coupling is $H_{ecl} = V \times \frac{V}{V_1} \times \frac{V}{V_2} \times \frac{2 \times S_{max}}{V}$. The expected effective coupling length for the net N_1 and N_2 is $ECL(N_1)=ECL(N_2) = V_{ecl}+H_{ecl}$.

4.2 Placement of Merging Points

Once the tree of segments has been built, we use a top-down approach to decide the exact merging points of each internal segment such that the overall expected coupling length among different clock trees is minimized. We decide the exact merging points of all clock trees at each level using a simulated annealing

approach. During the simulated annealing, the move decision is based on the change of ECL (cost) from the current point to the new point. When a point is moved along a merging segment, its corresponding ECL is recalculated based on the formula given in Section 4.1. The whole top-down process is summarized in Fig 7.

Procedure Top-Down Assignment

Initialize the merging point for all trees;
While bottom level of all trees has not been reached do
 for (i=0; i< Number of tree; i++)
  Extract segments of a tree i at CurrentLevel
 end for
 T = Initial Temperature
 While (T >= Minimum Temperature) do
  Move = 0;
  While (Move < Move_Per_Temperature)
   S = Pick One Segment Randomly at current level;
   P_{new} = Pick one point on S randomly;
   if ($ECL(P_{new}) < ECL(P_{old})$ or
    $P <= e^{Alpha(ECL\ New-Old)/T}$
   Move merging point on S from P_{old} to P_{new};
  End While
 End While
 Move one level down
End While

Fig. 7: Top-Down Assignment by Simulated Annealing

5. Minimal Crosstalk Two-Pin Net Router

The algorithm in section 4 produces a set of binary clock trees with assigned merging points. Each clock tree can be viewed now as a set of two-pin nets which have to be detail-routed. We define the minimal crosstalk routing problem as:

*Problem 2:***Minimal crosstalk routing problem:** Given a layout with some routed clock branches and a set of two-pin clock connections to be routed, the minimal crosstalk routing problem is to lay out the unfinished connections such that the total effective coupling length for each net is minimized.

5.1 Constrained Regions

According to the definition of effective coupling, any existing vertical wire segment introduces a region $2*S_{max}$ wide with the wire segment in its center. Similarly any existing horizontal wire segment introduces a region $2*S_{max}$ tall. Effective coupling occurs for neighboring wires only within those regions. An interconnect to be routed should avoid the regions spanned by the already routed wire segments. Consider a rectangle of width L and height $2*S_{max}$ introduced by a horizontal wire segment. Passing through this rectangle horizontally will add L to the effective coupling length, while passing it vertically will not contribute to the effective coupling length. We call such a rectangle an *X-constrained region*. Similarly, a vertical wire segment spans a *Y-constrained region*. Besides the X-constrained regions and the Y-constrained regions, there may exist areas occupied by both horizontal and vertical wires which introduce *XY-constrained regions*. If a wire crosses a particular constrained region it incurs a cost equal to the effective coupling length. Below, the regions are defined more precisely.

Definition 5: Constrained Regions: An *X-constrained region* is a region whose internal grids have the same unit x-cost (X_c) and a unit y-cost (Y_c) equal to zero. A *Y-constrained* region is a region whose internal grids have the same unit-y cost and a unit-x cost equal to zero. An *XY-constrained region* is a region whose grids have the same unit-x cost and the same unit-y cost. A *free region* is a region whose x-cost and y-cost are zero.

Each constrained region contains at least four boundaries, two horizontal boundaries (we call them *X-boundaries* hereafter), and two vertical boundaries (*Y-boundaries*)

5.2 Region Graph and Routing Path

We slice the routing plane into a set of constrained regions. Each region may have several neighboring regions which will share either an X-boundary or a Y-boundary. A region graph is defined as follows:

Definition 6: Region Graph: A mid-point on the X-boundary or Y-boundary of a region is a node in the region graph. There is an edge between two nodes if they correspond to the boundaries of the same region. Edges are assigned costs defined below. Fig. 8 shows an example of a routing region and its corresponding graph.

Definition 7: The *Cost of an Edge:* Consider two nodes $P_0(X_0,Y_0)$ and $P_1(X_1,Y_1)$ which belong to the same region, and let X_c, Y_c denote the unit costs of this region. The cost of an edge between the nodes corresponding to P_0 and P_1 in the region graph is the distance between P_0 and P_1 in terms of the unit costs in this region.

$$\varepsilon = |X_0 - X_1| \times X_c + |Y_0 - Y_1| \times Y_c$$

According to the definition of the regions, we observe that the routing path from one point to the other can be specified by a set of neighboring regions, or by a set of boundaries. The search for the minimal crosstalk path for a two-pin net is then transformed to finding a minimal cost path in the region graph from the source node to the destination node.

5.3 Crossing points assignment

As we have already observed, each possible routing path can be defined by a set of boundaries through which it passes. The problem is how to find such a crossing points assignment along a set of boundaries that the resulting path would be the shortest.

*Problem 3:***Minimum Cost Crossing Points Assignment:** Given a region graph, and a routing path specified as a set of boundaries, we would like to assign the path crossing points on each boundary such that the routing path has a minimum cost.

Definition 8: Minimum Escape Point: For a rectangle region specified by its corners (X_1,Y_1), (X_1,Y_2), (X_2,Y_1) and (X_2,Y_2); and a point P(X,Y) at the boundary or inside the region, there are four possible minimum escape points for P. They are (X,Y_1), (X,Y_2),

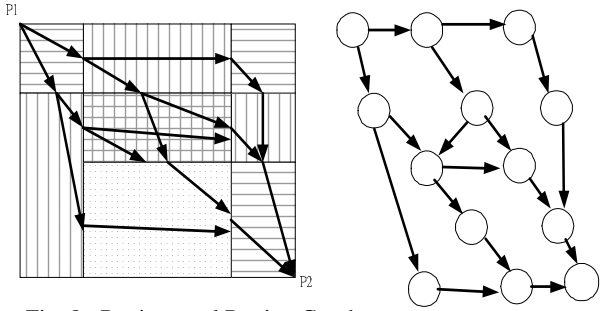
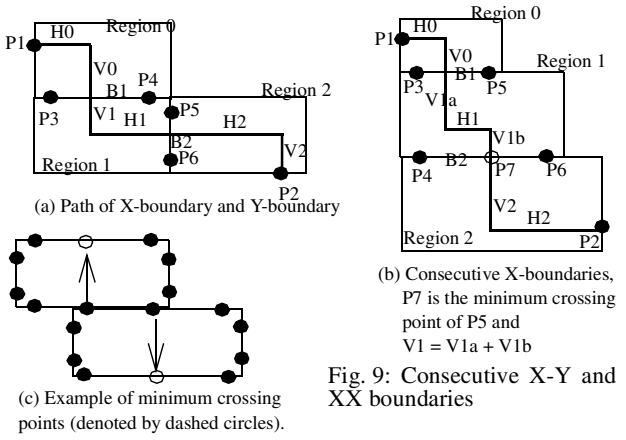


Fig. 8: Regions and Region Graph



(a) Path of X-boundary and Y-boundary

(b) Consecutive X-boundaries, P7 is the minimum crossing point of P5 and V1 = V1a + V1b

(c) Example of minimum crossing points (denoted by dashed circles).

Fig. 9: Consecutive X-Y and XX boundaries

(X_1,Y), and (X_2,Y) which are the projections of P on the boundaries of the rectangle.

Minimum escape points provide the shortest route for a point (including starting and destination points) to pass the region horizontally or vertically. Fig. 9(c) shows an example of possible minimum crossing points. In this figure solid circles represent end points of the boundaries. There are at least eight end points for each region. Transparent circles represent the minimum escape points.

Lemma 1: If a path passes through an X-boundary and then a Y-boundary, the minimal-cost crossing pin assignment occurs only at the end points of the X-boundary and Y-boundary.

Proof: Fig. 9(a) shows this configuration. We denote the unit x-cost of region 0 by X_{c0}, and unit y-cost by Y_{c0}. And X_{c1}, Y_{c1}, X_{c2}, and Y_{c2} are the unit x-costs and y-costs of the region 1 and region 2 respectively. The cost from P1 to P2 becomes

$$(H_0 \times X_{c0} + H_1 \times X_{c1} + H_2 \times X_{c2}) \quad \text{for horizontal cost,}$$

and $$(V_0 \times Y_{c0} + V_1 \times Y_{c1} + V_2 \times Y_{c2}) \quad \text{for vertical cost.}$$

A change of the pin assignment at the X-boundary will change only the ratio of H_0 and H_1, therefore will have not effect on the vertical cost. A change of the pin assignment on the Y-boundary will change only the ratio of V_1 and V_2, the horizontal cost will not be changed. $V_0 x Y_{c0}$ and $H_2 x X_{c2}$ are constant. So to minimize the horizontal cost from P_1 to P_2, H_0 must be assigned as small as possible if $X_{c0}>X_{c1}$. This implies that the point P_3 (the end point of B_1) is selected. If $X_{c0}<X_{c1}$, H_0 must be assigned as large as possible implying that P_4 (another end point of B_1) is selected. To minimize the vertical cost from P_1 to P_2, if $Y_{c1}>Y_{c2}$, P_5 (end point of B_2) must be chosen. Otherwise, if $Y_{c1}<Y_{c2}$, P_6 (another end point of B_2) must be selected. This lemma also holds for a path passing through a Y-boundary and then an X-boundary.

Lemma 2: If a path passes through two consecutive X-boundaries, the minimal cost crossing point assignment occurs at the end points of the two X-boundaries or at the minimum escape points.

Proof. Fig. 9(b) shows this configuration. The vertical cost $(V_0 \times Y_{c0} + V_1 \times Y_{c1} + V_2 \times Y_{c2})$ through these three regions is constant no matter how the horizontal segments are distributed. So to minimize the path cost requires minimizing the value of $(H_0 \times X_{c0} + H_1 \times X_{c1} + H_2 \times X_{c2})$. There are six possibilities in total: (1) $X_{c0}<X_{c1}<X_{c2}$, the minimum cost occurs at P_5 and P_6. (2) $X_{c1}<X_{c0}<X_{c2}$, the minimum cost occurs at P_3 and P_6. (3) $X_{c2}<X_{c0}<X_{c1}$, the minimum cost occurs at P_3, P_4. (4) $X_{c0}<X_{c2}<X_{c1}$, the minimum cost occurs at P_5, P_7. (5) $X_{c2}<X_{c1}<X_{c0}$, the minimum cost occurs at P_3, P_4. (6) $X_{c1}<X_{c2}<X_{c0}$, the minimum cost occurs at P_3, P_6. Lemma 2 also holds for two consecutive Y-boundaries.

Theorem 1: Given a path passing through a set of boundaries, the minimal-cost crossing points assignment occurs only at the end points of the passed boundaries or at the minimum escape points.

Proof: From Lemma 1, we already know that the theorem holds true for two different types of consecutive boundaries, X-boundary followed by a Y-boundary (type 1), or Y-boundary followed by an X-boundary (Type 2). The minimal-cost crossing point assignment occurs always at the end points of the X and Y-boundaries. From lemma 2, we know that for two consecutive boundaries of the same type, X-boundary followed by an X-boundary (Type 3), or Y-boundary followed by a Y-boundary (Type 4), the minimal cost crossing point assignment occurs always at the end points of the boundaries, or at the minimum escape points. A sequence of boundaries can be decomposed into these four types. Therefore, this theorem always holds for any combination of boundaries.

Theorem 1 limits the number of possible crossing points on the boundaries for the optimal path. It allows us to guide the path search on a plane by a solution to a simpler problem of finding a shortest path in a graph.

5.4 Modified Region Graph

The search for the minimal cost crosstalk path is performed on a modified region graph defined as follows.

Definition 9: Modified Region Graph: Let A and B be the two points to be connected. Each end point of region boundaries in the bounding box induced by A and B, the minimum escape points induced by those boundary points, and the points A and B, introduce a node in the modified region graph. There is an edge between two nodes if they correspond to the points in the same region. The edge cost is the same as in definition 7.

5.5 Algorithm

There are two steps. First we perform the shortest path algorithm on the modified region graph built for the points A and B to be connected. The result will be a route from A to B given as a sequence of boundaries and their crossing points. Then, we apply a two-point maze router on every consecutive pair of points found on the shortest path. The modified region graph is built only once for each clock tree, since we are concerned primarily with crosstalk among different clock trees.

6. Experimental Results

We have implemented our algorithm using GNU C++ running under a Linux operating system. Since there were no existing benchmarks for this problem, we took five randomly generated examples to test the algorithm. All the examples had two clocks with the same number of sink nodes. To compute ERC, a 2D polygon operation program [4] was used to calculate the intersections of routing regions. In the bottom-up phase, we used the program developed by [3] to solve the MCPM problem.

We supposed there were M clock signals, T_1, T_2,...,T_M, with routing areas A_1, A_2,...,A_M. First we measured the ratio of overlapping area with respect to total area $A_1+A_2+...A_M$. For comparison, we

TABLE 1. Ratio Of Area Overlapping

Ratio	#node	α=0.2 β=0.8	α=0.8 β=0.2	Dec%
Ex1024	1024	0.274	0.128	53.3%
Ex512	512	0.262	0.201	23.3%
Ex256	256	0.330	0.125	62.1%
Ex128	128	0.314	0.168	46.5%
Ex64	64	0.281	0.187	33.5%

used two sets of parameters as described in section 3.3, L(α=0.2 β=0.8), and C(α=0.8 β=0.2). The L parameter tends to minimize the wire length, and the C parameter tends to minimize the effective coupling. The bottom-up phase always significantly decreases the initial overlapping ratio, from 23% to 62% on the tested examples. The result is shown in Table 1.

TABLE 2. C_{couple}(C for α=0.8,β=0.2 L for α=0.2 β=0.8)

	L_{eff}	Length	C_cFF	Dec% L_{eff}	Cc
Ex1024/C	21533	113251	385	63%	71%
Ex1024/L	57454	113349	1326		
Ex512/C	21134	37693	62	30%	44%
Ex512/L	30121	38103	110		
Ex256/C	7908	26824	107	70%	60%
Ex256/L	13474	26275	265		
Ex128/C	2215	19324	22	72%	86%
Ex128/L	7789	18835	166		
Ex64/C	1247	13160	23	65%	76%
Ex64/L	3530	12958	99		

After the placement of merging points, we applied our two-pin net router to complete the layout. The total coupling capacitance was calculated after the clock trees were actually routed. C_{couple} is the summation of inter-clock capacitance and calculated based on 0.13um process characterized in Fig. 2. The result in table 2 shows about 44% to 86% decrease in coupling capacitance without significantly increasing the wirelength of the trees. After routing, identical buffers were inserted to the internal nodes of clock trees. To determine the maximum jitter, we fix the input rising edge of tree 1, and scan the possible input rising edge (positive coupling) as well as the falling edge (negative coupling) of tree 2. SPICE simulation is used to get the timing shift for each possible timing relationship. Table 3 shows the maximum jitter of tree 1. The result shows a 68% to 92% decrease of the maximum jitter. All the five examples are based on a 512x512 grid. Table 4 shows run time and memory requirement.

TABLE 3. Positive, Negative and Total Jitter

	Negative Shift(ps)	Positive Shift(ps)	Total Jitter (ps)	Dec%
Ex1024/C	44.96	31.13	76.09	74.1%
Ex1024/L	181.06	112.89	293.95	
Ex512/C	26.50	19.87	46.37	68.3%
Ex512/L	97.33	48.94	146.27	
Ex256/C	16.75	12.69	29.44	82.2%
Ex256/L	107.99	56.94	164.93	
Ex128/C	7.38	8.93	16.31	92.4%
Ex128/L	121.72	91.54	213.26	
Ex64/C	14.37	14.33	28.70	83.7%
Ex64/L	95.87	80.16	176.03	

TABLE 4. Run Time and Memory Requirement

	Run Time(sec)	Memory(MB)
Ex1024	726	27
Ex512	267	23
Ex256	114	19
Ex128	39	17
Ex64	7	15

7. Conclusion

In this paper we have described an efficient and effective algorithm for multi-clock tree routing to minimize jitter induced by capacitive clock coupling. We have formulated the problem as that of minimizing the effective coupling length between the trees, and we have demonstrated that such a cost function reduces the coupling capacitance between the routed trees. Our experimental results indicate that this cost function also captures jitter, as we were able to reduce jitter by 80% on the average if no shielding used. In case of shielding used, our algorithm can eliminate 60% of shielding requirement since the algorith can eliminate average 60% of effective coupling.

References

[1] T.H. Chao, Y.C. Hsu, J.M. Ho, K.D. Boese, and A.B. Khang, "Zero Skew Clock Routing With Minimal Wirelength", IEEE Trans. CAS, Vol.39, No.11, Nov 1992, pp.799-814.

[2] L.H Chen, and M. Marek-Sadowska, "Aggressor Alignment for Worst-Case Crosstalk Noise", IEEE Trans. Computer-Aided Design, Vol.20, No.5, May 2001, pp.612-621.

[3] W. Cook and A. Rohe, "Computing minimum-weight perfect matching", INFORMS J. on Computing, Vol. 11, 1999, pp. 138-148.

[4] K. Holwerda, "http://www.xs4all.nl/~kholwerd/bool.html"

[5] M.F. Hsiao, M. Marek-Sadowska, and S.J. Chen, "Crosstalk Minimization For Multiple Clock Tree Routing", Proc. IEEE MWSCAS, 2002.

[6] D. J.-H. Huang, A. B. Kahng, and C.-W. A. Tsao, "On the Bounded-Skew Clock and Steiner Routing Problems", Proc. ACM/IEEE Des. Aut. Conf., June 1995, pp. 508-513.

[7] A. Vittal, L.H. Chen, M. Marek-Sadowska, K.P. Wang, and S. Yang, "Crosstalk in VLSI Interconnections", IEEE Trans. Computer-Aided Design, Vol.18, No.12, Dec. 1999, pp.1817-1824.