VLSI DESIGN
1999, Vol. 10, No. 1, pp. 87–97
Reprints available directly from the publisher
Photocopying permitted by license only

© 1999 OPA (Overseas Publishers Association) N.V.
Published by license under
the Gordon and Breach Science
Publishers imprint.
Printed in Malaysia.

Lower-Power and Min-Crosstalk Channel Routing for Deep-Submicron Layout Design*

S. H. NAM[a], J. D. CHO[a,†] and D. WAGNER[b]

[a] *Dept. of Elect. and Comp. Eng., Sungkyunkwan Univ., Suwon, Korea 440-746;*
[b] *Fakultät für Math. und Informatik, Univ. Konstanz, Fach D 188, Konstanz, Germany D-78457*

(Received 7 September 1998; In final form 20 November 1998)

Consider a set of nets given by horizontal segments $S = \{s_1, s_2, \ldots, s_n\}$ and a set of tracks $T = \{t_1, t_2, \ldots, t_k\}$ in a channel, then a track assignment consists in an assignment of the nets to the tracks such that no two nets assigned to the same track overlap. One important goal is to find a track assignment with the minimum number of tracks such that the signal interference between nets assigned to neighboring tracks is minimized. This problem is called crosstalk minimization. For a given track assignment with k tracks, crosstalk can be reduced by finding another track assignment for S with k tracks (*i.e.*, by permuting tracks). However, considering all possible permutations requires exponential time. For general cost function for crosstalk measure, the problem is NP-hard. Several heuristic approaches were previously presented. In this paper, we consider special instances of the crosstalk-minimization problem where the cost function depends only on the length of the segments that runs in parallel and all pairs of segments intersect. An algorithm solving this problem in $O(n \log n)$ time is presented. An extension applied to the instances with more general function of switching activity and mixed signal sensitivity to reduce crosstalk and power consumption is also presented.

Keywords: Lower power design, VLSI layout, crosstalk, deep submicron design

1. INTRODUCTION

As CMOS technology advances into deep submicron, some of the net lengths for interconnection between modules can be so long that they have a wire resistance, R_{wire}, which is comparable to the resistance of the driver. As the interconnection delays become more and more dominant in the overall delay, performance-driven (for time and low power) routing becomes important, in addition to the traditional goals of area and interconnect minimization. The coupling capacitance, C_{coupling}, between minimum pitch wires on a 0.25 μm CMOS IC can account for over 80% of the total capacitance of a wire. This makes interconnect crosstalk noise one of the biggest challenges in VLSI design

*This work was supported by the Korean Science and Engineering Foundation under Grant 965-0918-005-2.
†Corresponding author. e-mail: jdcho@yurim.skku.ac.kr

today. When $R_{\text{wire}} \times C_{\text{coupling}} = t_{\text{rising}}$, there is noise problem. The increase of crosstalk not only holds for coupling *via* the interconnect, but also for the crosstalk *via* the substrate. The crosstalk is also proportional to the power consumption in CMOS circuits, thus minimizing crosstalk also leads to minimizing power consumption. A battery-operated multimedia system in 0.1 μm technology will require 40 NiMH battery cells which causes the system not to be portable.

Therefore, as the interconnection delays become more and more dominant in the overall delay, performance-driven routing becomes important, in addition to the traditional goals of area and interconnect minimization. Timing, routability, size, and power problems are not discovered until after detailed routing. Therefore, deep-submicron designs require crosstalk-free detailed routing. Major contributions on this paper is to develop a very fast optimal algorithm to improve the routing quality in terms of crosstalks for special instances of channel routing problems. One can extend the proposed basic algorithm framework to incorporate the other performance issues for practical use. Our work is significant and innovative since it is the first attempt at identifying the instances with a linear-time complexity to reduce the crosstalk in the channel routing problem.

1.1. Crosstalk Measure

Crosstalk is a capacious and inductive interference caused by the noise voltage developed on signal lines when nearby lines change state. It can occur in the following manner as in Figure 1. When the voltage of an aggressor signal changes as the voltage of a victim signal is in transition through the high gain region of a receiver circuit, a ripple or a small glitch is formed in the victim signal if the switching directions are opposite of each other. This high-gain crosstalk can affect a circuit in many different ways: In a static CMOS design, this glitch increases the wire delay considerably. It also increases the receiving circuit delay since it alters the effective input rise/fall time. This causes speed-related logic errors. Sometimes, this glitch can propagate through many fanout gates. If the signal feeds into a dynamic logic gate, it can discharge the storage charge during the evaluation phase and cause a logic error.

The crosstalk is a function of the separation between signal lines and the linear distance that signal lines run parallel with each other. To maximize system speed, crosstalk must be reduced to levels where no extra time is required for the signal to stabilize. Signals such as clocks, that are highly sensitive to crosstalk should be isolated by reference planes from signals on other layers and/or by extrawide line-to-line spacing.

Signals are grouped into categories based on waveshape control requirements, crosstalk limits, or other special requirements. For example, clocks, strobes, buses, memory address, data, chip-select, and write lines, and asynchronous signals, ECL signals, and analog signals have special routing requirements as follows [1].

- Data buses: Crosstalk between buses tends to be data-pattern-sensitive and is worse when all addresses or data lines change in the same direction at the same time.

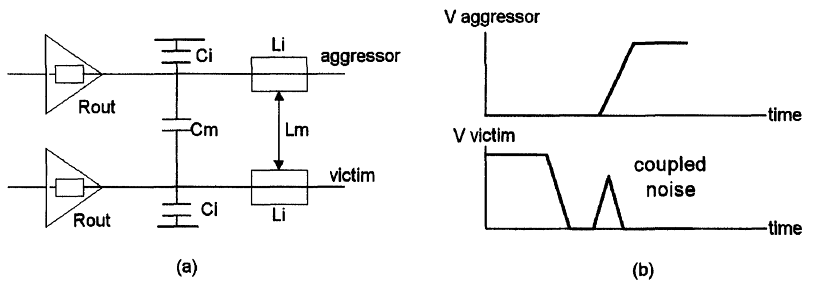


FIGURE 1 Crosstalk between two nets.

- Memory address and data signals: Cross-coupling between any combination of memory input or output signals as well as crosstalk between nearby unrelated signals can be disruptive and must be guarded against. Excessive cross-coupling from memory data lines to address lines during read cycles can result in positive feedback that degrades the response time of the memory device and in extreme cases can cause unstable oscillatory operation. That data-to-address-line cross-coupling may upset address lines sufficiently to cause write signals to incorrect memory locations. Memory wire lines require the highest possible degree of isolation from crosstalk. They must be isolated by reference planes and by extrawide line-to-line spacing from other signals, particularly other memory chip-selects, address line, and data buses. When signals are not isolated by planes, care must be taken to ensure that memory signals are not run directly above or below a critical signal for some length.
- Clock signals and Strobes: To meet crosstalk limits, clock signals must be isolated and confined between reference layers. Other signals must not be mixed with clocks. Clock signals on a given layer must have extra spacing between lines. Clock signals of different frequencies must have extrawide spacing, as well as clock signals and other signals if they are to be mixed.
- ECL and analog: ECL and analog signals require a high degree of isolation from TTL-or CMOS-level signals. They must be physically isolated in separate board areas with separate ground and voltage planes that are isolated from TTL or CMOS switching currents.

Each net in the mixed analog/digital circuits is identified depending upon its crosstalk sensitivity [2].

- Noisy = high impedance signal that can disturb other signals, *e.g.*, clock signals.
- High-Sensitivity = high impedance analog nets; the most noise sensitive nets such as the input nets to operational amplifiers.
- Mid-Sensitivity = low/medium impedance analog nets.
- Low-Sensitivity = digital nets that directly affect the analog part in some cells such as control signals.
- Non-Sensitivity = The most noise insensitive nets such as pure digital nets.

The crosstalk between two interconnection wires also depends on the frequencies (*i.e.*, signal activities) of the signals traveling on the wires.

Once the electrical designer has established the electrical requirements or limits of each signal category based on the system performance requirements and error budgets, the requirements must be translated to specific mechanical requirements for the routing people. For example, twisting signal lines with ground lines provides some shielding effects minimizing the chance for coupling into adjacent wiring. As the VLSI technology improves, more layers are available for routing. As a result, there is a need for developing multilayer routing scheme (*e.g.*, the layer assignment of 7-layer process is as follows. 1,2: local routing, 3,4: inter-block routing, 5,6: power, ground, top layer: clock) that reduces the die size (and thus the average wirelength) and crosstalks.

A number of papers have been published related on the crosstalk issues: mixed analog and digital applications [2]; crosstalk minimum layer assignment [3, 4]; a spacing algorithm [5]; a channel routing enhancement considering crosstalk by a linear programming of track permutations [6]. [7] also addressed a channel routing algorithm and also, [8] proposed an optimal algorithm for the problem of minimizing crosstalk between vertical wires in 3-layer VHV channel routing. Recently, a crosstalk-minimum rainbow k-color permutation based on left edge dynamic programming was presented by [9].

1.2. Power Measure

Power consumption in CMOS circuits is due to three sources: dynamic power consumption due to charging and discharging of capacitive loads during output transitions at gates, the short circuit current which flows during output transitions at gates, and the leakage current. The last two factors above can be made sufficiently small with proper device and circuit design techniques, thus, research in design automation for low power has focused on minimization of the first factor, the dynamic power consumption.

The average dynamic power P_{av} consumed by a CMOS gate is given below, where, C_l is the load capacity at the output of the node, V_{dd} is the supply voltage, T_{cycle} is the global clock period, N is the number of transitions of the gate output per clock cycle, C_g is the load capacity due to input capacitance of fanout gates, and C_w is the load capacity due to the interconnection tree formed between the driver and its fanout gates.

$$P_{av} = 0.5\frac{V_{dd}^2}{T_{\text{cycle}}} C_l N = 0.5\frac{V_{dd}^2}{T_{\text{cycle}}}(C_g + C_w)N \quad (1)$$

Logic synthesis for low power attempts to minimize $\sum_i C_{g_i} N_i$ whereas physical design for low power tries to minimize $\sum_i C_{w_i} N_i$. Here C_{w_i} consists of $C_{x_i} + C_{s_i}$, where C_{x_i} is the capacitance of net i due to its crosstalk, and C_{s_i} is the substrate capacitance of net i.

For lower power layout applications, power dissipation due to crosstalk is minimized by ensuring that wires carrying high activity signals are placed sufficiently far from the other wires. Similarly, power dissipation due to substrate capacitance is proportional to the wirelength and its signal activity.

We need to minimize $C_{x_i} + C_{s_i}$ to both minimize crosstalk and power consumption. In this paper, we address an effective algorithm on the crosstalk minimization problem. An extension applied to a layout with minimum power consumption is also presented.

This paper is organized as follows. We formulate the problem in Section 2. In Section 3, we present an optimal algorithm for the crosstalk minimization problem in the special case of channel routing. Sections 4 and 5 will present experimental results and conclusion, respectively.

2. FORMULATION OF THE PROBLEMS

In a channel, given a set of multi-terminal nets N specified by the locations of their terminals on two channel sides, top layer is usually reserved for vertical wires and the bottom layer is reserved for horizontal wires. The general 2-layer crosstalk-minimum channel routing problem is known to be NP-complete.

The complexity of the channel routing stems from the vertical constraints. The vertical constraints imply that the two nets whose pins are at the same row in the channel cannot be overlapped vertically. Two-layer channels are usually dense and crosstalk-sensitive. Thus, it is crucial to attain the desired crosstalk minimization solution.

In early 90's, a third metal layer became feasible. Most of the current gate-array technologies use three layers for routing. For example, the Motorola 2900ETL, DEC's Alpha chip, Intel's 486 chip used a three metal layer process and original Intel Pentium was also fabricated on a similar process. The three-layer routing algorithm can be classified into two main categories: the reserved layer and the unreserved layer model. The reserved layer model can further be classified into the VHV model and the HVH model. Note that in VHV routing, the vertical constraints between nets no longer exist. Therefore, the channel height which is equal to the maximum density can always be realized using Left-Edge-Algorithm. Without vertical constraints, more nets are permutable in a channel. Thus, at a cost of one more layer, the VHV routing is effective in terms of both area and crosstalk.

There are pairs of nets that cannot be assigned to the adjacent tracks because some nets might strongly interfere each other. Note that we can reduce crosstalk by maximizing the track separation between pairs of nets with high crosstalk. Thus, we formulate the crosstalk minimization problem in VHV model as follows.

Definition 1 Given k tracks $T = (t_1, t_2, \ldots, t_k)$ and n intervals, *i.e.*, horizontal segments of net, $S = (s_1, s_2, \ldots, s_n)$, $k \le n$. The *crosstalk minimization problem* is to find an assignment ϕ of the intervals to the tracks, *i.e.*, $\phi : S \to T$, such that no two intervals assigned to the same track intersect, and the *cost function*

$$\sum_{(s_i,s_j)\in S} w_{i,j} \textit{ subject to } |\phi_{s_i} - \phi_{s_j}| = 1 \qquad (2)$$

is minimum, where w_{ij} is $X_{ij} \times (N_i + N_j) \times A_{ij}$: N_i(resp. N_j) = switching activity of net i (resp. net j); A_{ij} = signal sensitivity (for mixed signal interactions) between nets i and j; $X_{ij} = L_{ij}/D_{ij}$ = coupled noise between nets i and j; L_{ij} = coupled length between horizontal wires of nets i and j; D_{ij} = separative distance between horizontal wires of nets i and j.

A special case of the crosstalk minimization problem where $n = k$ is *track permutation* for crosstalk minimization.

Definition 2 Given an assignment of $S = (s_1, s_2, \ldots, s_k)$ to $T = (t_1, t_2, \ldots, t_k)$, such that each track t_i is assigned exactly one net s_j. Find a permutation of the nets $\Pi(S) = (\pi_{s_1}, \pi_{s_2}, \ldots, \pi_{s_n})$, *i.e.*, π_{s_j} is the track number assigned to s_j, such that

$$\sum_{(s_i,s_j)\in S} w_{i,j} \textit{ subject to } |\pi_{s_i} - \pi_{s_j}| = 1. \qquad (3)$$

is minimum. We denote such an optimal ordering by Π_{opt}.

Note that in the worst case the crosstalk is proportional to $(N_i + N_j)$. For example, when all addresses or data lines change in the same direction at the same time.

Let us identify horizontal segments of nets in the channel with intervals. Then an *interval clique* is a set of intervals whose corresponding interval graph is a clique. That is, when scanning the channel from left to right say, we consider the sets of intervals corresponding to all the intervals intersecting a vertical cut-line as depicted in Figure 2.

In the following, we are only concerned with crosstalk-minimum track permutation. In Section 3, we will describe an algorithm considering only X_{ij}, called the *first order model*, in Section 3.1, and extend the algorithm considering $X_{ij}(N_i + N_j)A_{ij}$, called the *second order model*, in Section 3.2.

2.1. Traveling Salesman Problem (TSP)

Instance Set S of n cities, distance $w(s_i, s_j) \in \mathcal{Z}^+$ for each pair of cities $s_i, s_j \in S$, positive integer B.
Question Is there a tour (hamiltonian cycle) of S *i.e.*, a permutation $< \pi(s_1), \pi(s_2), \ldots, \pi(s_n) >$ of S, whose edges' weight sum is no longer than B.

The problem is NP-complete in general graphs. A brute-force algorithm generates $n!$ tours and a dynamic programming uses $O(n^2 2^n)$ time.

TSP is a special case of crosstalk-minimization for interval clique, provided an arbitrary cost function is given. So crosstalk-minimization provided the cost function is arbitrarily chosen is NP-hard, even in the interval clique case. The first order model we consider is easier not because of restriction to interval clique, but because of our restriction to a very special cost function, *i.e.*, just the length of the interferences.

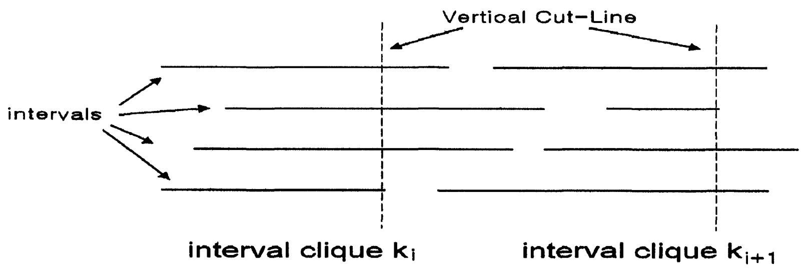


FIGURE 2 Two interval cliques in a channel.

2.2. Algorithm on the First-order Crosstalk Model

Consider an interval clique $S = \{s_1, s_2, \ldots, s_n\}$, where $s_i = (\ell_i, r_i)$, $\ell_i \leq r_i$, where ℓ_i and r_i represents x-coordinates of the left and right end points of the interval s_i. The length $L(I)$ of an interval $s = (\ell, r)$ is defined as the quantity $|r - \ell|$. A simple heuristic to the problem of finding minimum-crosstalk track assignment on interval cliques is to adapt a "greedy" algorithm.

ALGORITHM Greedy (Interval Clique)

- *Step 1* *assigned* = *Null*; *unassigned* = S; s_0 = a virtual segment corresponding to top channel shore; s_{n+1} = a virtual segment corresponding bottom channel shore;
- *Step 2* Select two segments s_i and s_j from unassigned such that w_{ij} is the largest, and assign s_i to t_1 and s_j to t_2; assigned = $\{s_i, s_j\}$; unassigned = $S - \{s_i, s_j\}$;
- *Step 3* Select a segment s_k such that crosstalk gain, when segment k is inserted between two segments (**Case 1**) 0 and i, or (**Case 2**) between i and j, or (**Case 3**) between j and $n+1$, $g(ij, k) = w_{ij} - (w_{ik} + w_{kj})$ is maximized. Then
  - for **Case 1**, assign s_k to t_1, s_i to t_2, and s_j to t_3;
  - for **Case 2**, assign s_i to t_1, s_k to t_2, and s_j to t_3;
  - for **Case 3**, assign s_i to t_1, s_j to t_2, and s_k to t_3;

  assigned = *assigned* + $\{s_k\}$; *unassigned* = *unassigned* − $\{s_k\}$
- *Step 4* Repeat Step 2 until all segments are inserted to the position with the most gain.

Even though the approach generates an optimal solution in most of instances, the time complexity of the Greedy Algorithm is $O(n^3)$ which may be not practical for large n. Also, we do not know yet whether the approach yields an optimal solution. We show in the next paragraph that there exists a polynomial-time algorithm to the crosstalk-minimum track permutation problem on an interval clique.

An interval clique can be partitioned into two subsets $S_\in$ and $S_\otimes$ as follows.

PROCEDURE Clique-Partition(Input S; Output $S_\in, S_\otimes$)

- *Step 1* Consider a vertical cut-line that intersects all intervals in S, and for $s_i \in S$ denote *left*(s_i) the part of s_i to the left of that cut line and *right*(s_i) the part of s_i to the right of that cut line. Accordingly, partition the interval clique into two sets S^{left} and S^{right}.
- *Step 2* Sort the intervals in S^{left} in increasing order of their wire length, and let $S^{\text{left}}_{\text{short}}$ contain the $\lfloor n/2 \rfloor$ shortest intervals from S^{left} and $S^{\text{left}}_{\text{long}}$ the $n - \lfloor n/2 \rfloor$ longest intervals from S^{left}. We denote by $\tau(s_i) \in \{\text{SHORT}, \text{LONG}\}$ the type of the interval $s_i \in S^{\text{left}}$. Then, $\tau(s_i \in S^{\text{left}}_{\text{short}}) = \text{SHORT}$ and $\tau(s_i \in S^{\text{left}}_{\text{long}}) = \text{LONG}$.
- *Step 3* Apply the process in Step 2 to S^{right}.
- *Step 4* $S_\in := \{s_i \in S : \tau(\mathit{left}(s_i)) = \tau(\mathit{right}(s_i))\}$; $S_\otimes := \{s_i \in S : \tau(\mathit{left}(s_i))(\neq \tau(\mathit{right}(s_i))\}$;

We call $S_\in$ *Containment Interval Clique* (*CIC*) and $S_\otimes$ *Monotone Interval Clique* (*MIC*). The following properties of $S_\in$ and $S_\otimes$ are easy to see.

COROLLARY 1 *Let the cardinality of S be n, and the cardinalities of $S_\in$ and $S_\otimes$ be $n_\in$ and $n_\otimes$ respectively.*

1. *The number of intervals $s_i \in S_\in$ with $\tau(\mathit{left}\ (s_i)) = \tau(\mathit{right}(s_i)) = \text{SHORT}$ is $\lfloor n_\in/2 \rfloor$.*
2. *$S_\in$ can be partitioned in two sets $S_1 := \{s_i \in S_\in : \tau(\mathit{left}(s_i)) = \tau(\mathit{right}(s_i)) = \text{SHORT}\}$ and $S_2 := \{s_i \in S_\in : \tau(\mathit{left}(s_i)) = \tau(\mathit{right}(s_i)) = \text{LONG}\}$ such that each interval from S_1 is contained in all intervals from S_2.*
3. *$n_\otimes$ is always even, where the number of intervals $s_i \in S_\otimes$ with $\tau(\mathit{left}(s_i)) = \text{LONG}$ is $n_\otimes/2$.*
4. *$S_\otimes$ can be partitioned into two sets $S_1 := \{s_i \in S_\otimes : \tau(\mathit{left}(s_i)) = \text{LONG}\}$ and $S_2 := \{s_i \in S_\otimes : \tau(\mathit{left}(s_i)) = \text{SHORT}\}$, where $|S_1| = |S_2|$.*

For an interval $s_i \in S_\in$ *let* $\tau(s_i) := \text{SHORT}$ *if* $\tau(\mathit{left}\ (s_i)) = \tau(\mathit{right}(s_i)) = \text{SHORT}$ and $\tau(s_i) := \text{LONG}$ otherwise.

ALGORITHM 1 Track Permutation on a Containment Interval Clique $S_\in$

- *Step 1* Partition
  After partitioning as in Step 2 of Procedure Clique-Partition,
  $S_1 := \{s_i \in S_\in : \tau(s_i) = \text{SHORT}\}$
  $S_2 := \{s_i \in S_\in : \tau(s_i) = \text{LONG}\}$.
- *Step 2* Assignment
  Let $T = \{t_1, t_2, \ldots, t_n\}$ be the set of tracks. Assign each $s_i \in S_1$ to distinct even number track t_{2m}, and each s_i of S_2 to distinct odd number track t_{2m+1}.

LEMMA 1 (Algorithm 1) *Generates a track permutation for a Containment Interval Clique with minimum crosstalk in time $O(n \log n)$.*

Proof The track assignment generated by Algorithm 1 is an alternate LONG-SHORT sequence (refer to Fig. 3). When $n_\in$ is odd, the lower bound on crosstalk is $\sum_{i=1}^{\lfloor n_\in/2 \rfloor} 2L(s_i)$. Similarly, the lower bound on crosstalk when $n_\in$ is even is $\sum_{i=1}^{n_\in/2-1} 2L(s_i) + L(s_{n_\in/2})$. It is obvious that crosstalk for the alternate LONG-SHORT sequence meets the above lower bounds. The time complexity is dominated by sorting the interval lengths. ■

COROLLARY 2 *Consider a track assignment generated by Algorithm 1 for a Containment Interval Clique $S_\in$. Then the crosstalk for a track assignment induced by a permutation on S_1 and S_2 respectively is again minimum.*

Consequently, we can resolve a vertical constraint (*i.e.*, the case where two vertical wire segments overlap) without increasing the crosstalk, by exchanging two intervals having end points of the same x-coordinate.

ALGORITHM 2 Track Permutation on a Monotone Interval Clique $S_\otimes$

- *Step 1* Partition each interval in $S_\otimes$ into two sets (left and right sets) using a vertical cut line that intersects all intervals in $S_\otimes$. We denote the left part of an interval s_i as *left*(s_i) and the right part as *right*(s_i). Let $S_\otimes^{\text{left}}$ (resp. $S_\otimes^{\text{right}}$) denote the set of intervals containing all *left*(s_i) (resp. *right*(s_i)). Then, both $S_\otimes^{\text{left}}$ and $S_\otimes^{\text{right}}$ are considered as a Containment Interval Clique.
- *Step 2* Apply Algorithm 1 to $S_\otimes^{\text{left}}$.

LEMMA 2 (Algorithm 2) *Algorithm 2 generates a track permutation for a Monotone Interval Clique with minimum crosstalk in $O(n \log n)$ time [refer to Fig. 4].*

Proof Algorithm 1 applied to $S_\otimes^{\text{left}}$ generates a track permutation where the intervals from $S_\otimes^{\text{left}}$ form an alternate LONG-SHORT sequence. Corollary 1 induces that $S_\otimes^{\text{right}}$ also forms an alternate LONG-SHORT sequence. ■

COROLLARY 3 *Consider a track assignment generated by Algorithm 2 for a Monotone Interval Clique $S_\otimes$. Then the crosstalk for a track assignment induced by a permutation on $S_1 = \{s_i \in S_\otimes : \tau(left(s_i)) = LONG\}$ and $S_2 = \{s_i \in S_\otimes : \tau(left(s_i)) = SHORT\}$ respectively is again minimum.*

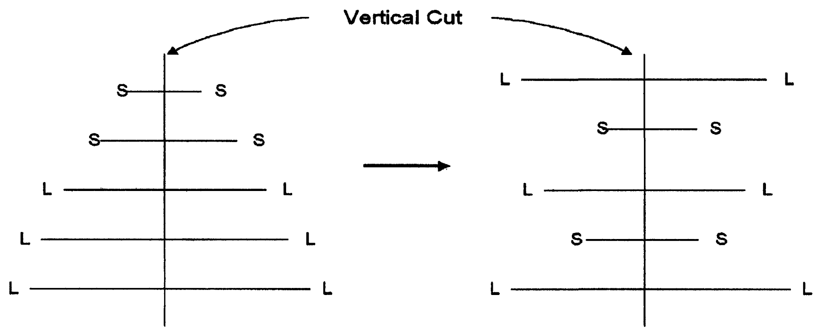


FIGURE 3 Algorithm 1 on a containment interval clique.

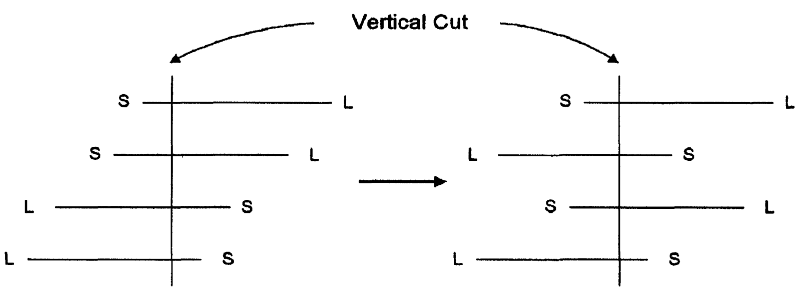


FIGURE 4 Algorithm 2 on a monotone interval clique.

We can satisfy a vertical constraint (*i.e.*, the constraint that two vertical wire segments are not allowed to overlap) by exchanging two intervals that cause the vertical constraint.

Based on the above two lemmas, we have the following algorithm for a general Interval Clique.

ALGORITHM 3 Track Permutation on an Interval Clique S

- *Step 1* Apply Clique-Partition (S; $S_\in$, $S_\otimes$).
- *Step 2* Apply Algorithm 1 for $S_\in$ and Algorithm 2 for $S_\otimes$.
- *Step 3* Find Merge-Clique.

PROCEDURE Merge-Clique(Input Track Assignment for $S_\in$ and $S_\otimes$ Respectively; Output Track Assignment for S)

Case A $|S_\in| = ODD$:

- *Step 1* Identify s_1, s_2, s_3 and s_4 such that

$$L(s_1) = \min_{s_i \in S_\in}\{L(\text{right}(s_i)) | \tau(\text{right}(s_i)) = \text{LONG}\}$$
$$L(s_2) = \min_{s_i \in S_\otimes}\{L(\text{right}(s_i)) | \tau(\text{right}(s_i)) = \text{LONG}\}$$
$$L(s_3) = \max_{s_i \in S_\otimes}\{L(\text{right}(s_i)) | \tau(\text{right}(s_i)) = \text{SHORT}\}$$
$$L(s_4) = \max_{s_i \in S_\in}\{L(\text{right}(s_i)) | \tau(\text{right}(s_i)) = \text{SHORT}\}$$

Permute track assignment generated by Algorithm 1 for $S_\in$ such that intervals s_1 and s_4 are on neighboring tracks. Permute track assignment generated by Algorithm 2 for $S_\otimes$ such that s_2 is uppermost and s_3 lowermost interval of $S_\otimes$. Form track assignment for S by inserting $S_\otimes$ into $S_\in$ between s_1 and s_4 (refer to Fig. 5a).

- *Step 2* Identify s_1, s_2, s_3 and s_4 such that

$$L(s_1) = \min_{s_i \in S_\in}\{L(\text{left}(s_i)) | \tau(\text{left}(s_i)) = \text{LONG}\}$$
$$L(s_2) = \min_{s_i \in S_\otimes}\{L(\text{left}(s_i)) | \tau(\text{left}(s_i)) = \text{LONG}\}$$
$$L(s_3) = \max_{s_i \in S_\otimes}\{L(\text{left}(s_i)) | \tau(\text{left}(s_i)) = \text{SHORT}\}$$
$$L(s_4) = \max_{s_i \in S_\in}\{L(\text{left}(s_i)) | \tau(\text{left}(s_i)) = \text{SHORT}\}$$

Permute track assignment generated by Algorithm 1 for $S_\in$ such that intervals s_1 and s_4 are on neighboring tracks. Permute track assign-

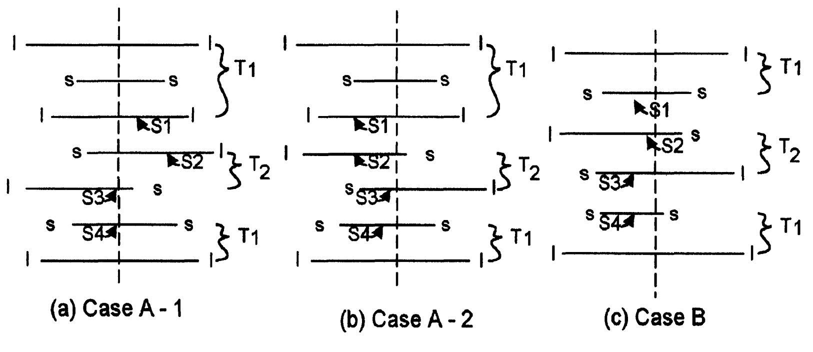


FIGURE 5 Procedure merge-clique.

ment generated by Algorithm 2 for $S_\otimes$ such that s_2 is uppermost and s_3 lowermost interval of $S_\otimes$ and then flip track assignment of $S_\otimes$. Form track assignment for S by inserting $S_\otimes$ into $S_\in$ between s_1 and s_4 (refer to Fig. 5b).

- *Step 3* Select S with minimum crosstalk from Step 1 and Step 2.

Case B $|S_\in|$ = EVEN:

- Identify s_1, s_2, s_3 and s_4 such that

$$L(s_1) = \max_{s_i \in S_\in}\{L(\text{left}(s_i)) | \tau(\text{left}(s_i)) = \text{SHORT}\}$$
$$L(s_2) = \max_{s_i \in S_\otimes}\{L(\text{left}(s_i)) | \tau(\text{left}(s_i)) = \text{SHORT}\}$$
$$L(s_3) = \max_{s_i \in S_\otimes}\{L(\text{right}(s_i)) | \tau(\text{right}(s_i)) = \text{SHORT}\}$$
$$L(s_4) = \max_{s_i \in S_\in}\{L(\text{right}(s_i)) | \tau(\text{right}(s_i)) = \text{SHORT}\}$$

Permute track assignment generated by Algorithm 1 for $S_\in$ such that intervals s_1 and s_4 are on neighboring tracks. Permute track assignment generated by Algorithm 2 for $S_\otimes$ such that s_2 is uppermost and s_3 lowermost interval of $S_\otimes$. Form track assignment for S by inserting $S_\otimes$ into $S_\in$ between s_1 and s_4 (refer to Fig. 5c).

LEMMA 3 (Merge-Clique) *Procedure Merge-Clique yields an optimal merging.*

Proof Note that after clique-partition the number of intervals in $S_\otimes$ is always even. Thus, we have three cases as in Figure 5. In Figure 5a, at left-hand side of vertical cutline we have an alternate LONG-SHORT sequence, thus the case is out of our concern. At right-hand side of vertical cutline, $\tau(right(s_1)) = \tau(right(s_2))$ = LONG. Intervals s_1 and s_2 are chosen such that crosstalk between s_1 and s_2 is minimized. Similarly, $s_3 \in S_\otimes$ and $s_4 \in S_\in$) are chosen such that crosstalk between s_3 and s_4 is minimized. The proof on Case 2 in Figure 5b is similar. In Figure 5c, the number of intervals in $S_\in$ and $S_\otimes$ are both even. Thus, we find only LONG-SHORT and SHORT-SHORT sequence at both left-hand and right-hand side of vertical cut-line as in Figure 5c. ■

THEOREM 1 (Algorithm 3) *Algorithm 3 generates a track permutation for an Interval Clique with minimum crosstalk in O(n log n) time.*

Proof Finding optimal solutions for both $S_\in$ and $S_\otimes$ individually takes $O(n \log n)$ time. The Merge-Clique operation takes $O(n)$ time. ■

2.3. Extension to the Second Order Crosstalk Model

As described in Section 2, Crosstalk in the second order model is proportional to $X_{ij}(N_i + N_j)A_{ij}$. The algorithm for the second order model is as follows. We first partition the intervals into 4 interval groups as shown in Figure 6, and then apply the

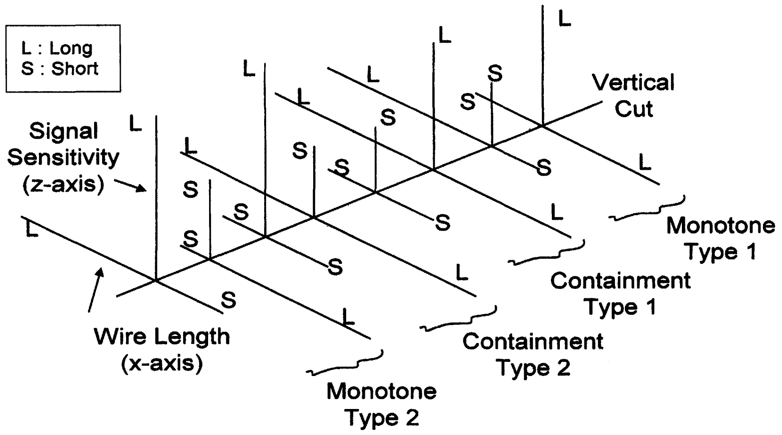


FIGURE 6 The second-order crosstalk modeling.

TABLE I A comparison result

trs	Crosstalk avg.			Wire length avg.			CPU sec		
	L.E	Our	B.F	L.E	Our	B.F	L.E	Our	B.F
5	31	24	24	81	80	80	0.02	0.03	1
7	63	48	48	155	151	151	0.03	0.06	73
9	55	34	34	167	148	148	0.05	0.09	60,000
11	86	53	N/A	249	217	N/A	0.06	0.10	N/A
21	339	207	N/A	895	751	N/A	0.15	0.28	N/A

trs: number of tracks; L.E: Left-Edge algorithm;
Our: Algorithm 3; B.F: Brute Force Alg.

procedure for the first order model. Note that by partitioning the intervals as in Figure 6, the instance of the second order model consists of four smaller instances of the first order model. This is true because we no longer need to consider the signal sensitivity part. For each partitioned interval group, we apply Algorithm 1 or Algorithm 2 according to its type. Then, we apply Algorithm 3 to merge those four partitioned interval cliques. For brevity, we omit the detailed algorithm description.

3. EXPERIMENTAL RESULTS

We have experimented our algorithm (Algorithm 3) using SUN Ultra-Sparc 2 and Pentium-Pro Machine with C/C++. We compared our algorithm with left-edge algorithm, and also with brute-force method which generates an optimal enumerative solution. Table I shows the results obtained by using our algorithm, left-edge algorithm, and brute-force method for the track permutation problem on an interval clique, respectively. We tested each algorithm in 10,000 times by using random generated interval cliques. For using brute-force method on the examples which have more than 9 tracks as Table I, we cannot get a result due to the exponential running time. For the cases which have less than 9 tracks, our algorithm generates the same result as the brute-force method. For all cases, the average crosstalk is about 30.

4. CONCLUSION

In this paper, we consider special instances of the crosstalk-minimization problem where the cost function depends only on the length of the segments that runs in parallel and all pairs of segments intersect. An algorithm solving this problem in is presented. An extension applied to the instances with more general function of switching activity and mixed signal sensitivity to reduce crosstalk and power consumption is also presented. The presented algorithm can be applied to a performance-driven lower power channel routing in deep submicron VLSI designs.

Acknowledgements

I (Jun-Dong Cho) would like to thank Prof. Yoji Kajitani at Tokyo Institute of Technology for his inviting me to his laboratory for discussing many VLSI layout problems including the crosstalk minimization problem. The authors wish to acknowledge the financial support of the Korea Research Foundation and DFG for the Cooperative Research under The Korean-German Cooperative Science Program during 1997–1998.

References

[1] Buchanan, J. E. (1996). "Signal and Power Integrity in Digital Systems", McGraw Hill.

[2] Harada, I., Kitajawa, H. and Kaneko, T. (1990). "A Routing System for Mixed A/D Standard Cell LSI's", In: *International Conference on Computer-Aided Design*, pp. 378–381.

[3] Cho, J. D., Raje, S., Sarrafzadeh, M., Sriram, M. and Kang, S. M. (1993). "Crosstalk Minimum Layer Assignment", In: *Proc. IEEE Custom Integr. Circuits Conf., San Diego, CA*, pp. 29.7.1–29.7.4.

[4] Chen, H. H. and Wong, C. K. (1996). High Performance Design Automation for MCM and Packages, chapter "63-Layer TCM Wiring with Three-Dimensional Crosstalk Constraints", World Scientific, Cho, J. D. (Ed.), Franzon, P. D. (Co-Editor), pp. 81–92.

[5] Chaudhary, K., Onozawa, A. and Kuh, E. (1993). "A Spacing Algorithm for Performance Enhancement and Crosstalk Reduction", In: *International Conference on Computer-Aided Design*, pp. 697–702.

[6] Gonzalez, T. F. and Lee, S. (1993). "Routing Around Two Rectangles to Minimize the Layout Area", *Algorithmic Aspects of VLSI Layout, World Scientific*, Sarrafzadeh, M. and Lee, D. T. (Eds.), pp. 365–397.

[7] Kirkpatrick, D. A. and Sangiovanni-Vincentelli, A. L. (1994). "Techniques for Crosstalk Avoidance in the physical design of High-Performance Digital Systems", In: *International Conference on Computer-Aided Design*, pp. 616–619.

[8] Thakur, S., Chao, K.-Y. and Wong, D. F. (1995). "An Optimal Layer Assignment Algorithm for Minimizing Crosstalk for Three Layer VHV Channel Routing", to appear in VLSI DESIGN, *an International Journal of Custom-Chip Design*, Simulation, and Testing, Jun-Dong Cho (Guest Editor).

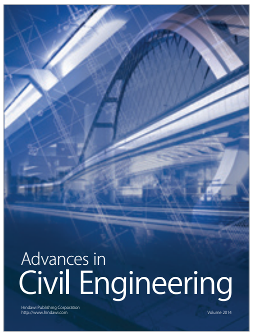
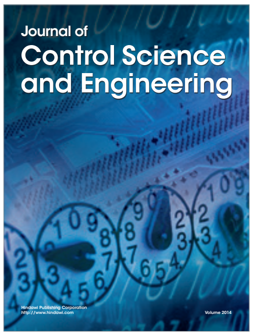
[9] Cho, J. D. and Chang, M. S., "LEXA: A Left-Edge based Crosstalk-Minimum *k*-Colour Permutation in VHV Channels", *ISCAS*, pp. 1704–1707, June 1997.

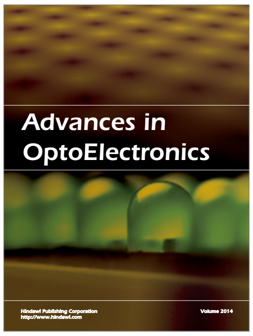
Authors' Biographies

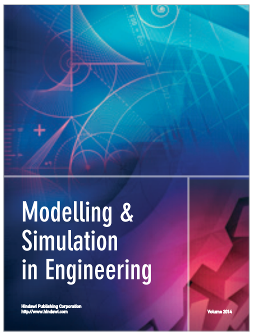
Jun-Dong Cho is an associate Professor of Electrical and Computer Engineering at Sungkyunkwan University, Korea. He received the Best paper award at the 1993 *Design Automation Conference* in the area of physical design. He has been serving on a guest editor of *VLSI DESIGN* for the special issue in *Design Automation for VLSI Interconnects* and, he has a book *High-Performance Design Automation of MCMs and Packages*, World Scientific, 1996, and the co-author of an invited chapter in Encyclopedia of Electrical and Electronics Engineering in the area of VLSI Circuit Layout. This is published in 1999 by John Wiley and Sons, Inc. His research interests include VLSI Computer Aided Design algorithms and graph theory applications. He is a Senior member of the IEEE, and Member of ACM.

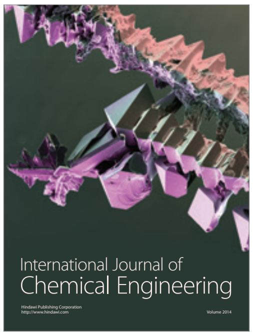
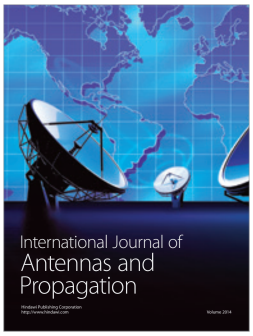
Su-Hyun Nam has received his Master degree in Electrical and Computer Engineering at Sungkyunkwan University in Suwon, Korea. His research interests include VLSI Computer Aided Design algorithms.

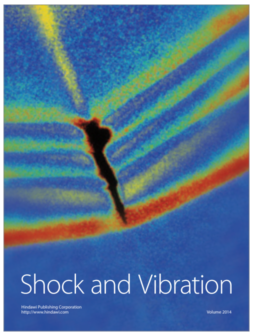
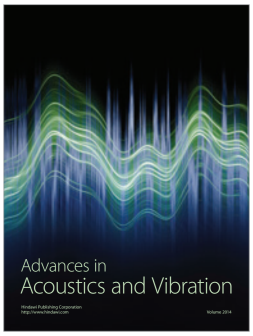
Dorothea Wagner is a full Professor of Computer Science at the Universitaet Konstanz. She is coauthor of a chapter on "Disjoint paths in planar graphs" in the DIMACS-Series in Discrete Mathematics and Theoretical Computer Science, Volume on the "Year of Combinatorial Optimization" and of a chapter on "VLSI Network Design" in the "Handbooks in Operations Research and Management Science", Volume on "Network Routing". Her research interests include discrete optimization and graph algorithms particularly applied to VLSI-design, transportation systems and visualization. She is member of the Editorial Board of Journal of Graph Algorithms and Applications.





Hindawi

Submit your manuscripts at
http://www.hindawi.com