Timing- and Crosstalk-Driven Area Routing

Hsiao-Ping Tseng, *Member, IEEE*, Louis Scheffer, *Senior Member, IEEE*, and Carl Sechen, *Member, IEEE*

***Abstract*—We present a timing- and crosstalk-driven router for the chip assembly task that is applied between global and detailed routing. Our new approach aims to process the crosstalk and timing constraints by ordering nets and tuning wire spacing in a quantitative way. The new approach fits between global routing and detailed routing along the physical design flow. It is the first to address the timing- and crosstalk-driven area routing problem using crosspoint assignment prior to the detailed routing stage, in contrast to the most previous approaches applied in the post-detailed routing stage. Our new approach enjoys a larger optimization solution space than the previous approaches whose solution space is highly limited by routed geometric constraints. Based on the global routing information, our graph-based optimizer preroutes wires on the global routing grids incrementally. The graph-based optimizer has two stages, net order assignment and space relaxation. A quick capacitance extraction and Elmore delay calculator considering signal switching activities are implemented to find the timing of critical nets and to provide the timing slack database of critical nets. As the graph-based algorithm proceeds, the path delay of critical nets and the timing slack database are updated. During the optimization process, it only optimizes the timing critical paths with negative slack values. The experimental results show a 5%–16% delay reduction for MCNC macrocell benchmark circuits for a 0.25-μm process for wire geometric ratio (height/width) = 1.0, against a 25% delay reduction if there is infinite space around each metal wire on the same layer. It shows a remarkable 8.4%–25% delay reduction for MCNC benchmarks for wire geometric ratio = 2.0, against a 33% delay reduction if there is infinite space around each metal wire. As experimental results indicate, our new approach overall achieves an average 46% reduction on intralayer coupling capacitance. Therefore, the delay reduction by our optimizer is more significant when the minimum feature size continuously shrinks and the wire geometric ratio increases.**

***Index Terms*—Area routing, crosstalk, net order, timing driven.**

I. Introduction

A. Overview

One of the emerging issues in deep submicrometer technology is the increasing effect of coupling capacitance between interconnect on the same layers. Our capacitance extractor shows that for typical process parameters the intra-layer coupling capacitance contributes 25% of metal2 interconnect capacitance in a 0.25-μm process if the wire geometric ratio (height/width) is 1.0 and 33% if the wire geometric ratio is 2.0. As the feature size shrinks, the wire geometric ratio becomes larger and wires have less spacing to neighboring wires. Wire resistance and intralayer coupling capacitance therefore increase as feature sizes shrink. It has the direct impact that the interconnect delay exceeds the gate delay in deep submicrometer technology. Most of the existing routing frameworks only consider wire length as the constraint for path delay and ignore the neighboring switching correlation. Previous work in the crosstalk-aware routing problem [7]–[17] is mostly in the gridded domain in which spacings between wires are fixed. However, in many cases, a deep submicrometer routing framework should no longer route signals at minimum wire width and at minimum spacing. None of the previous work can optimize interconnect delay by tuning net ordering and adjusting spacing in a quantitative way at a global level.

The interconnect coupling capacitance causes signal integrity problems in two respects, timing faults and logic faults. In Fig. 1(a), the coupling capacitance between signals I1 and I2 causes interference in several ways. If two signals switch at the same time in the same direction, the coupling capacitance does not affect the voltage level of the two signal lines at least to first order and we say the effective neighboring coupling capacitance $C_{\text{eff}} = 0$. If one signal switches and the other is silent, the effective neighboring coupling capacitance $C_{\text{eff}} = C$. If two signals switch at the same time in the different directions, they affect each other and increase the delay such that $C_{\text{eff}} = 2C$. In a general case, the neighboring coupling capacitance ranges from $0C$ to $2C$ and depends on the switching correlation between two coupled wires. For a static gate, the switching activities of signal I1 may cause pulses on the neighbor signal I2 [see Fig. 1(c)]. The crosstalk pulse may be passed to subsequent gate levels, causing unneeded signal transitions and therefore increased power dissipation. For a dynamic gate, suppose the output I1 has a coupling capacitance to the input $m2$ of gate $g2$, as shown in Fig. 1(d). During the evaluation period ($\varphi = 1$), the upward switching of signal I1 causes an upward glitch on the input $m2$ of dynamic gate $g2$, which can result in a nonrestorable loss of charge at the storage node I2, thus propagating a wrong value to the next stage. There are many heuristics and techniques, such as wire permutation, buffer insertion, and buffer sizing, which can decrease coupling capacitance or improve timing and potentially reduce logic fault hazards and timing fault hazards. In this paper, we mainly concentrate on improving the worst case timing.

To estimate the crosstalk effect from circuit-switching activities, our path-delay calculation uses a switching factor (SF) with regard to the timing (transition) windows of two signals. If the transition windows of two signals overlap but switch in the same direction, then for the best case the $\text{SF} = 0$ and the effective coupling capacitance $C_{\text{eff}} = 0C$ (for the worst case, $\text{SF} = 1$ and $C_{\text{eff}} = C$). If one signal switches and the other

Manuscript received November 30, 1997; revised June 25, 1999 and December 13, 2000. This work was supported by the Semiconductor Research Corporation, Digital Equipment Corporation, and Sun Microsystems. This paper was recommended by Associate Editor M. Pedram.

H.-P. Tseng and C. Sechen are with the Department of Electrical Engineering, University of Washington, Seattle, WA 98195 USA.

L. Scheffer is with Cadence Design Systems Inc., San Jose, CA 95134 USA.

Publisher Item Identifier S 0278-0070(01)01943-1.

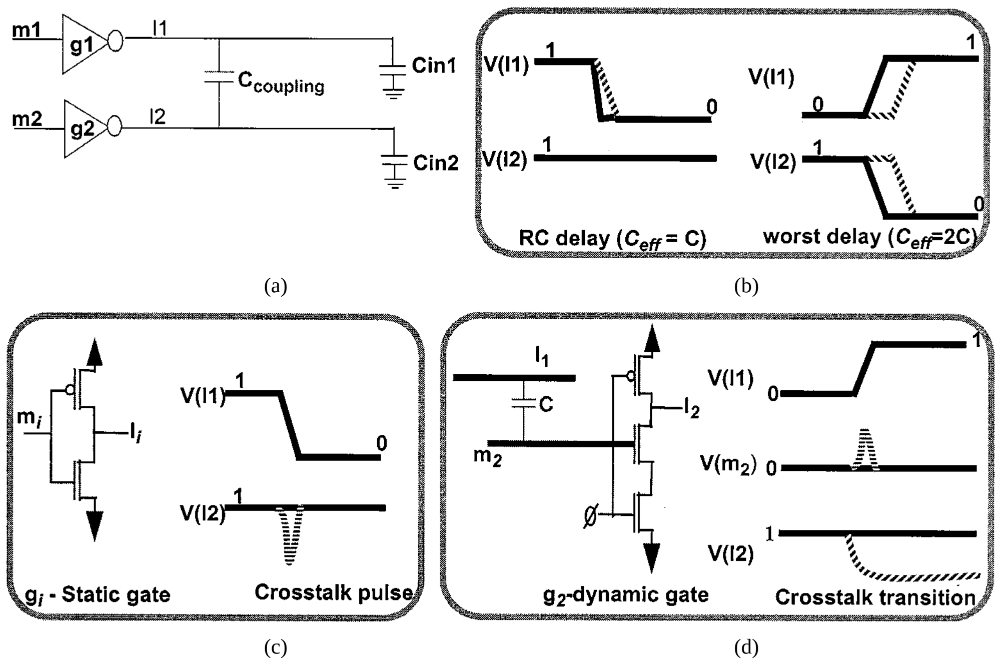


Fig. 1. Crosstalk effect (the dashed lines are signal waveforms due to C_{eff}). (a) Basic circuit. (b) Timing fault. (c) Logic fault-static gate. (d) Logic fault-dynamic gate.

is silent, then for both the best and worst case the switching factor $\text{SF} = 1$ and $C_{\text{eff}} = 1C$. Conversely, if the transition windows of two signals overlap but switch in different directions, the $\text{SF} = 2$ and $C_{\text{eff}} = 2C$. Previous work [5] on predicting circuit switching activities has proved that it is very difficult and inaccurate to guess switching activities from circuit functionality. In this paper, we assume the switching factors of signals are provided by synthesis tools or the user.

B. Previous Work in Crosstalk Reduction Routing

In this paper, we shall present a new method to adjust net ordering and wire spacing in the post-global routing stage by assigning the positions of cross points along the boundaries of routing regions. To compare with the previous approaches in the crosstalk reduction domain, we briefly give overview of the previous work in this section. Previous crosstalk reduction work falls into three stages along the design flow, post-global routing, detailed routing, and post-layout spacing.

1) Post-Global Routing: Xue *et al.* [6] developed a post-global routing crosstalk risk analyzer that estimates the possible coupling between sensitive nets and tries to reroute nets away from crosstalk risky zones. Its graph-based approach solves the problem nicely, but lacks a convincing crosstalk estimation without a detailed routing. The crosstalk reduction heuristic at the post-global routing stage has the great freedom to optimize the layout in a larger solution space than other work done in the detailed routing stage.

2) Detailed Routing: Crosstalk reduction efforts have been made at a detailed routing level for the river routing problem [8], the switchbox problem [13], and the channel routing problem (CRP) [9]–[12]. However, these algorithms are all in the net ordering domain with fixed spacing in which segments are rearranged during the postprocessing stage such that accumulated coupling capacitance of critical nets is reduced. The segment rearrangement heuristics of the above approaches only apply to nets in a local scope, such as in a switchbox or in a channel.

The crosstalk-driven routing problem for mixed-signal or analog circuits have been approached by [14]–[16]. The gridless graph-based channel router by Dubois and Donzelle (D&D) [15] is an extension to the Glitter channel router [3]. The D&D channel router uses precalculated augmented spacing rules under the consideration of crosstalk constraints, contrary to the minimum spacing rule in Glitter. Kirkpatrick's crosstalk-driven channel router [9] is mainly extended from Glitter and he proves the problem to be in the NP-complete domain. His routing framework has a more sophisticated mechanism to transform the crosstalk budget of sensitive nets into topological constraints (spacing rules) in a constraint graph than the D&D channel router. Malavasi [14] developed a gridded area router which transforms the measures of local crowding, resistance, and capacitance into the weight set of the A^* searching algorithm. If timing constraints in the final routing have been violated, weights are modified based on the knowledge of performance sensitivities and of the contribution of the associated parasitics to constraint violations. The circuit is routed again with the new weight set at up to a finite number of iterations.

3) Post-Layout Spacing Heuristic: Chaudhary *et al.* [17] proposed a post-layout graph-based spacing algorithm in 1993. The framework spaces out wires after detailed routing has been finished, contrary to the prerouting strategy of our framework prior to detailed routing. Our framework is different from Chaudhary's work in two aspects—the measure of the crosstalk effect and the graph-based algorithm. Chaudhary's measure of the crosstalk effect is based on crosstalk voltage glitches instead of actual delay in our framework. In his work, the crosstalk effect from driver to sink is simply the superposition of all glitches of wires along the path. The cost function is, therefore,

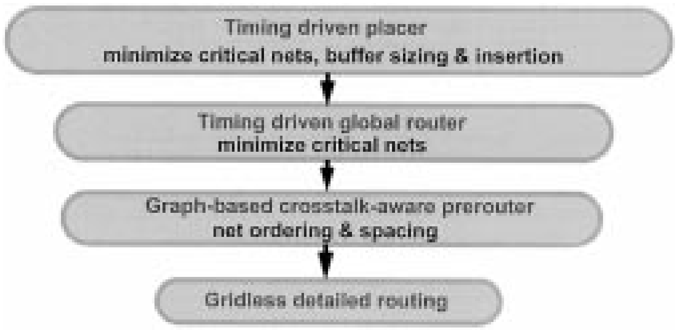

Timing driven placer
minimize critical nets, buffer sizing & insertion
Timing driven global router
minimize critical nets
Graph-based crosstalk-aware prerouter
net ordering & spacing
Gridless detailed routing

Fig. 2. Timing-driven place and route flow.

more for the measure of logic fault hazards rather than timing fault hazards in our approach. However, it more or less over estimates the crosstalk glitches because it does not consider the resistance–capacitance (*RC*) filter effect on a path from driver to sink and directly sums up all the unweighted glitches on a path. Chaudhary's graph-based algorithm selects the node that has the worse coupling and increases the edge weight between them. The potential disadvantage of this greedy operation is the misutilization of space resources around a timing critical wire segment compared with our new spacing heuristic, which will be presented in Section VI.

C. Overview of Our Crosstalk-Driven Area Routing Approach

For industrial circuits, critical nets tend to be long and go across multiple global routing cells—either channels or switch-boxes. Fig. 4 shows an example of a chip assembly circuit in which most nets go across multiple global routing cells. It is necessary to calculate the timing slack value on a path-delay basis, instead of on a local G cell delay basis. Therefore, a good crosstalk optimizer must work at a global level and consider the path delay from driver to sink.

We shall propose a practical and efficient approach that adjusts net ordering and relaxes wire spacing in a global scope and on a path-delay basis. The new approach works like a prerouter and fits between global routing and detailed routing in a timing-driven place and route flow, as shown in Fig. 2.

In the first stage of our prerouter, the net order algorithm constructs a constraint graph to represent all the segments in the circuit. It assigns proper net orders starting from the most congested global routing cell boundary toward the least congested global routing cell boundary. In the second stage, the space relaxation algorithm allocates available spaces to the regions around critical nets according to their timing requirements. A quick capacitance extractor and an Elmore delay calculator are implemented to update the slack database of path delays along the whole optimization process. Only the timing critical signals with negative slack are optimized along the incremental process. The flow of the crosstalk-aware prerouter is depicted in Fig. 3.

The remainder of the paper describes our new timing- and crosstalk-driven area prerouter as follows. The problem we shall solve is formulated in Section II and is proved to be NP-complete. We propose a new graph-based framework to solve the problem in Section III. The *RC* extraction model and interconnect delay calculation are described in Section IV. The net ordering algorithm and the space relaxation algorithm are presented in Sections V and VI, respectively. The experimental results are shown in Section VII. We shall draw a conclusion in Section VIII.

II. Problem Formulation

We aim to optimize the timing of critical nets in the presence of crosstalk. We focus on large chip assembly circuits on a global level. The algorithm is integrated between global routing and detailed routing and used as a prerouter for the detailed router. We describe the interesting problem in Section II-A and formulate it in Section II-B. The proof of NP completeness of this problem is presented in Section II-C.

A. Chip Assembly Area Routing Problem

The chip assembly routing problem is too complex to solve in one piece and is often solved by divide-and-conquer approaches—global routing, then detailed routing. The layout floorplan is first broken onto a three-dimensional (3-D) global routing plane of global cells (G cells). A global router searches for the rough path of each signal on the global cell plane and determines into which global cells each signal should traverse. Based on the global routing information, a detailed router is able to efficiently complete the routing of one G cell at a time. From the practice of industrial chip assembly circuits [see the example in Fig. 4(a)], we observe that most signals traverse through multiple Gcells, preferably by straight segments. To achieve the goal that long nets are straightened, net ordering optimization must be done on sets of G cells instead of inside a single G cell. The ordering of nets and spacing between them can be determined by the crosspoints on G cell boundaries if all wires are straightened [see Fig. 4(b)]. Chip assembly is clearly a case in which crosspoint assignment can determine the layout almost as well as the final routing. Kao *et al.* [4] develop a crosspoint assignment algorithm to improve the routability but it does not consider timing and crosstalk. We define the crosspoint assignment problem, considering timing and crosstalk constraints in the next section.

B. Timing-Driven Crosspoint Assignment Problem

In this paper, we shall solve the crosspoint assignment problem by a graph-based framework. The crosspoint of a signal determines the position of two segments connected to the crosspoint. In Fig. 5, crosspoint $c1.1$ of net 1 determines the y value of segments $s1.1a$ and $s1.1b$. If crosspoints $c1.1$ and $c1.2$ of net 1 have different values, a jog appears between segment $s1.1b$ and $s1.2a$. Redundant jogs take out extra routing resources and thus increase the routing difficulty. To maximize the detailed routability for each global cell, the number of jogs should be minimized. For example, crosspoints $c1.1$ and $c1.2$ are suggested to have the same value. The left end of segment $s1.1.a$ is determined by crosspoint $c4$ on the neighboring layer.

The crosspoint assignment problem that we aim to solve includes several optimization objectives—routability, crosstalk effect, and path delay. In our delay calculation, we have considered the signal switching activities along with the interconnect coupling capacitance. Therefore, the main optimization objectives fall into two groups—routability and path delay. We

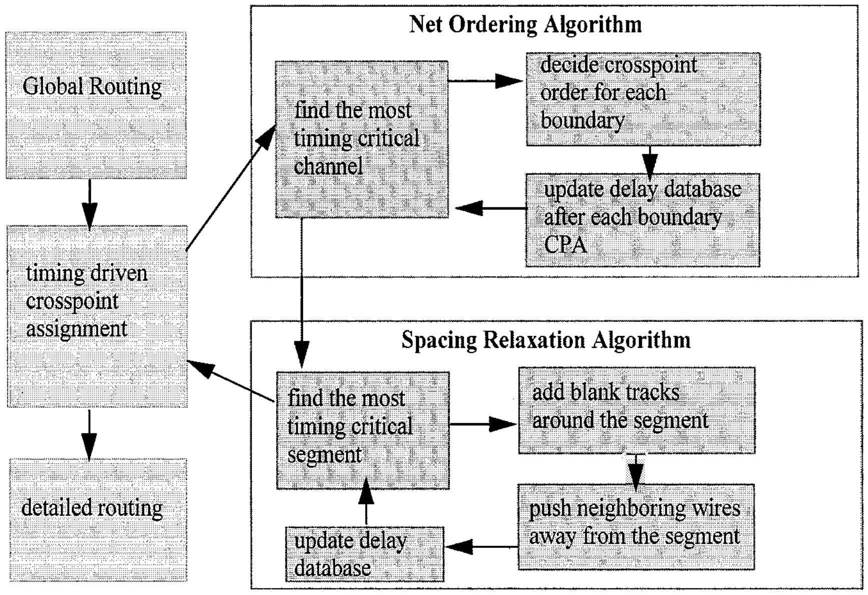

Global Routing
timing driven crosspoint assignment
detailed routing
Net Ordering Algorithm
find the most timing critical channel
decide crosspoint order for each boundary
update delay database after each boundary CPA
Spacing Relaxation Algorithm
find the most timing critical segment
add blank tracks around the segment
push neighboring wires away from the segment
update delay database

Fig. 3. Basic flow of crosspoint assignment heuristic.

call the problem of assigning crosspoints on G cell boundaries to optimize routability and path delay to be the timing-driven crosspoint assignment (TDCA) problem, described as follows.

Given the G cell boundary $G_g \in$ routing channel $R = \{G_{g0}, G_{g1}, \ldots\}$ with top boundary $G_{\text{TOP}g}$ and bottom boundary $G_{\text{BOT}g}$, a global routing of net n is $\{seq_{n,0}, seq_{n,1}, \ldots\}$, where $seq_{n,s}$ is the sth sequence of consecutive crosspoints in the same channel from the driver. Crosspoint $c_{n,s,i} \in \{c_{n,s,0}, c_{n,s,1}, \ldots\} \in seq_{n,s}$ has attached segments $\{s_{n,s,i,\mathbf{a}}, s_{n,s,i,\mathbf{b}}, s_{n,s,i,\mathbf{c}}\}$, where $s_{n,s,i,\mathbf{a}}$ is the segment to the left, $s_{n,s,i,\mathbf{b}}$ the segment to the right, and $s_{n,s,i,\mathbf{c}}$ the floating jog next to $s_{n,s,i,\mathbf{b}}$ if applicable. A directed path from the driver to sink j: $P_{n,j} = \{c_{n,s,i}|$ in the ascending order of its distance from the driver through the global routing path; $seq_{n,s}$ and $c_{n,s,i}$ are on the global routing path to sink $j\}$.

Objective:

1) path delay for $P_{n,j}$ through segments $\{s_{n,s,i,\mathbf{a}}, s_{n,s,i,\mathbf{b}}, s_{n,s,i,\mathbf{c}}|seq_{n,s}$ and $c_{n,s,i} \in P_{n,j}\}$ meets the timing constraint;
2) number of jogs $\{s_{n,s,i,\mathbf{c}}|c_{n,s,i\in P_{n,j}}\}$ is minimized.

Subject to:

1) position $\text{pos}(c_{n,s,i}) < G_{\text{TOP}}$ and $\text{pos}(c_{n,s,i}) > G_{\text{BOT}}$;
2) $\text{pos}(c_{n1,s1,i1}) - \text{min\_dist} \geq \text{pos}(c_{n2,s2,i2})$ or $\text{pos}(c_{n1,s1,i1}) + \text{min\_dist} \leq \text{pos}(c_{n2,s2,i2})$, where min_dist is the minimum distance between net $n1$ and net $n2$; $(n1, s1, i2) \neq (n2, s2, i2)$.

In Section II-C, we shall show that objective 1 is NP-complete. Objective 2 is in fact a CRP and, therefore, NP-complete.

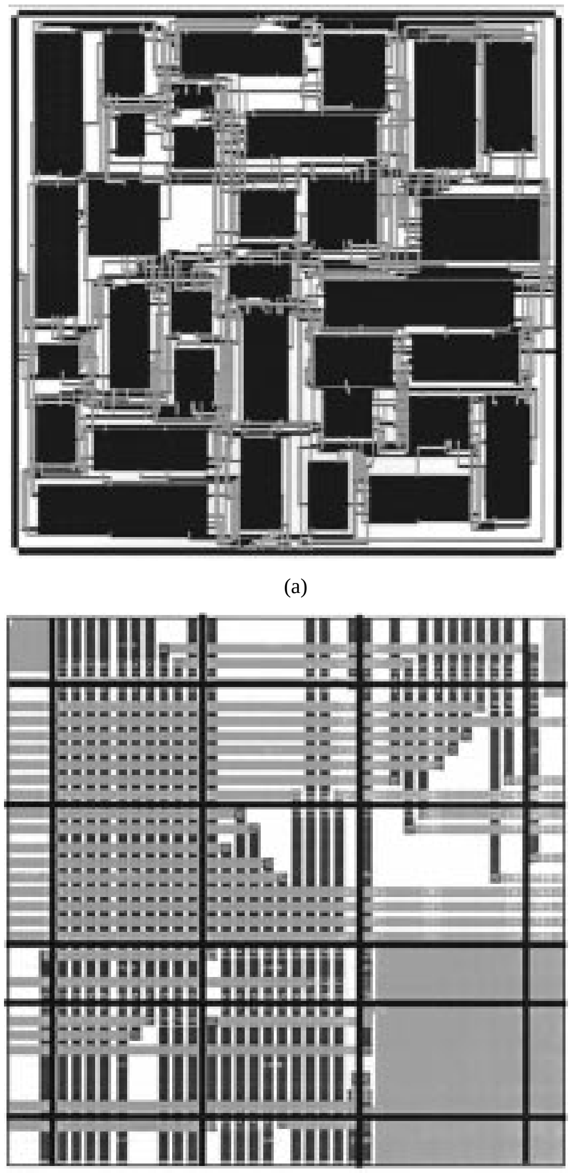
C. NP Completeness of TDCA Problem

The TDCA problem determines the crosspoint ordering (wire ordering) and crosspoint spacing (wire spacing). We shall show that the fix-spacing subset of the TDCA problem is NP-complete. We simplify the TDCA problem without loss of generality to have only one single G cell boundary with a constant spacing between each pair of crosspoints. This proof is based on the minimum crosstalk wire ordering (MCWO) problem in [10]. Let the signals going through the boundary be $\{net_0, \ldots, net_{N-1}\}$. The crosstalk $C(i, j)$ of edge (i, j) between net i and net j is equal to the effective coupling capacitance $S_{ij}C_{ij}$, where S_{ij} is the switching factor and C_{ij} is the coupling capacitance between net i and net j. According to the proof of the MCWO problem in [10], an $N \times N$ crosstalk matrix $[C]$, where $C(i, j)$ specifies the crosstalk between net i and net j, is given as distance matrix (i, j) of the traveling salesperson problem. Finding the minimum total crosstalk is reduced from finding the maximum Hamiltonian path cost in $[C]$. Therefore, the single G cell boundary fix-spacing TDCA problem is NP-complete.

However, the delay of a timing path is not a linear weighted sum of *RC* of segments on the path. Finding the optimal crosspoint ordering with the minimum *RC* on one single G cell boundary is not the same as finding the optimal timing delay for the whole routing channel or the whole circuit. Note that the TDCA problem also needs to deal with vertical constraints in the routing channel to maximize the routability for detailed router. The CRP is NP-complete which makes finding optimal results for the TDCA problem even more difficult.

III. Frame Work—Physical Layout

On the meshed global routing cell basis, we merge all of the global routing cells on the same column or the same row into a *vertical channel* or a *horizontal channel*, respectively. All of the wires or obstacles within a channel are represented by nodes inside a constraint graph. Therefore, the ordering of segments and spacings between them can be naturally determined by the edges between nodes. Our two algorithms—the net ordering algorithm and the space relaxation algorithm are based on this constraint graph framework. The definitions of nodes for movable segments and blockages are described in Section III-A. We shall explain how to calculate a node's physical location from the edges within the graph in Section III-B. Putting everything



(a)

(b)

Fig. 4. (a) Chip assembly circuit. (b) Detailed routing on a meshed G cell plane (solid lines are the G cell boundaries).

together, we shall transform the crosspoint assignment problem into a fixed channel routing problem (FCRP) in Section III-C.

A. Node and Graph

A crosspoint c_i determines the $y(x)$ value of its two connected horizontal (vertical) segments s_{i_a} and s_{i_b}. Let crosspoint c_i's position $\text{pos}(c_i) = y$ if its connected segments are horizontal, otherwise $\text{pos}(c_i) = x$. The signal of the net through crosspoint c_i is $sig(c_i)$. A floating node represents a movable straightened segment. Floating node $n = \{c_i | 0 < i < C, \text{pos}(c_i) = P, sig(c_i) = S\}$ represents a set of crosspoints with the same signal appearing in consecutive boundaries at the same position. In Fig. 6, if c_{1_0} and c_{1_1} are represented by different nodes, a jog is needed to connect segments $s_{1\_0\_b}$ and $s_{1\_1\_a}$. If c_{1_0} and c_{1_1} are represented by a node, we say segments $\{s_{1\_0\_a}, s_{1\_0\_b}, s_{1\_1\_a}, s_{1\_1\_b}\}$ are straightened. From the practice in routing compact circuits, we observe that straightened wires take less routing resources and therefore improve routability. In our framework, the most important effort to improve routability is to straighten wires with the same signal as much as we can within a channel.

If there is a hard pin pin_h inside channel C, the crosspoints having the same signal as pin_h in channel C are merged to form a fixed node which is stuck to the hard pin (see hardpin pin_h and c_h in Fig. 6). Blockages (prerouted wires or blocks) are represented by *fixed nodes* in the graph (see fixed node X_0 and X_1 in Fig. 6). Top node T_c represents the top boundary of channel c and bottom node B_c represents the bottom boundary of channel c. The ordering and spacing between nodes are described in the next section.

B. Edge and Weight

The *directed* edge $d(i, j)$ from node i to node j indicates node i to be above node j. The weight $wt(i, j)$ of edge between node i and node j is equal to the center-to-center distance between two nodes. The *undirected* edge $e(i, j)$ between node i and node j indicates that the two nodes are to be at least $wt(i, j)$ apart. The set of directed edges is denoted as E_{directed}. Movable segments (floating nodes) are constrained inside their local channels. Floating node F_n inside channel c has an edge $d(T_c, F_n)$ and an edge $d(F_n, B_c)$. Node F_{c+1} inside the upper channel $c+1$ whose horizontal span overlaps with that of F_n is connected to F_n by a directed edge $d(F_{c+1}, F_n)$ [see net 3 and net 1 in Fig. 6, F_3 and F_1 in Fig. 7(b)]. Node F_{c-1} inside the lower channel $c-1$ which overlaps with F_n is connected to F_n by an edge $d(F_n, F_{c-1})$ [see net$_h$ and net 4 in Fig. 6, X_h and F_4 in Fig. 7(b)]. The layout in Fig. 6 is represented by a graph in Fig. 7(b).

Node j is an *ancestor* of node i if $d(j, i)$ exists or a directed path $P = \{d(j, p_0), d(p_0, p_1), \ldots, d(p_{N-1}, i)\}$ from node j to node i exists. Node j is a descendent of node i if $d(i, j)$ exists or a directed path $P = \{d(i, p_0), d(p_0, p_1), \ldots, d(p_{N-1}, j)\}$ from node i to node j exists. An *ancestor chain* is a set of nodes on a directed path from the top node to node i, and a *descendent chain* is a set of nodes on a directed path from node i to the bottom node. The ancestor weight (ANCW) and descendent weight (DESW) of a node are defined as follows.

1) ***Top Node:*** $\text{DESW}_{Tc} = \text{ANCW}_{Tc} =$ the y (or x) value of the top boundary of channel c, if channel c is a row (column) channel.
2) ***Bottom Node:*** $\text{DESW}_{Bc} = \text{ANCW}_{Bc} =$ the y (or x) value of the bottom boundary of channel c, if channel c is a row (column) channel.
3) ***Fixed Node:*** $\text{DESW}_{Xn} = \text{ANCW}_{Xn} =$ the y (or x) value of the center of fixed node X_n.
4) ***Floating Node:***

$$\begin{aligned} \text{ANCW}_{Fn} &= \min(\text{ANCW}_i - wt(i, F_n)) \\ &\quad d(i, F_n) \in E_{\text{directed}} \\ \text{DESW}_{Fn} &= \max(\text{DESW}_j + wt(F_n, j)) \\ &\quad d(F_n, j) \in E_{\text{directed}}. \end{aligned}$$

Clearly, ANCW_i is the upper bound of node i's position if node i and its ancestor chains are fully compacted toward

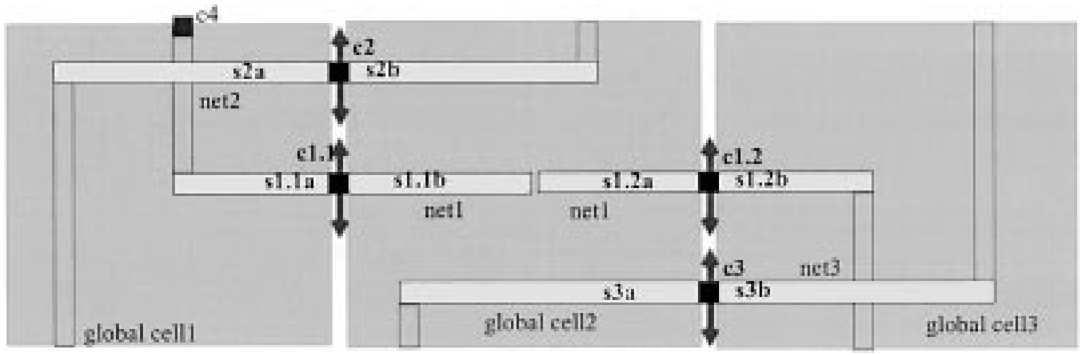


Fig. 5. Wiring models.

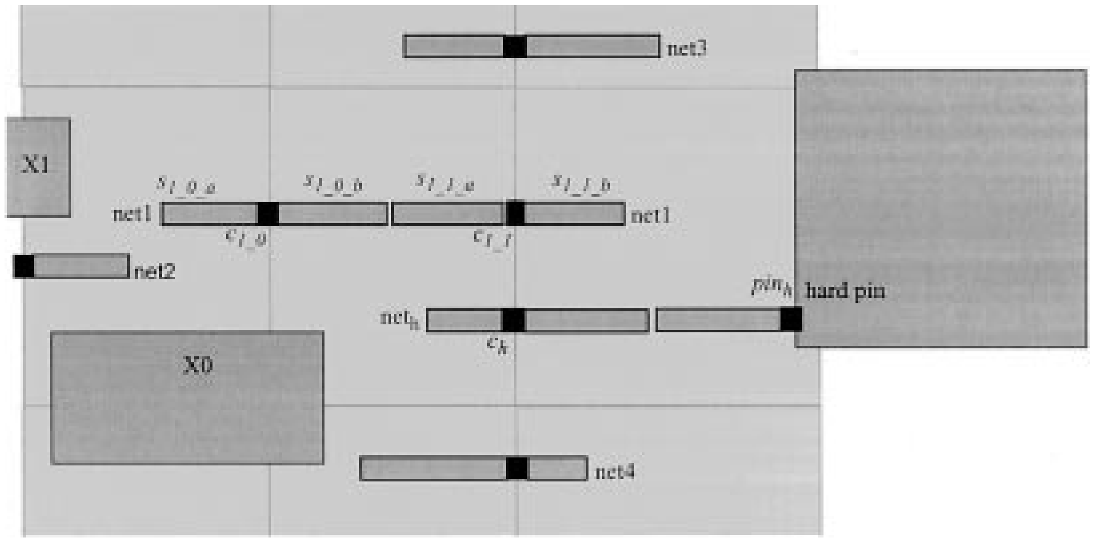


Fig. 6. Floating node and fixed node.

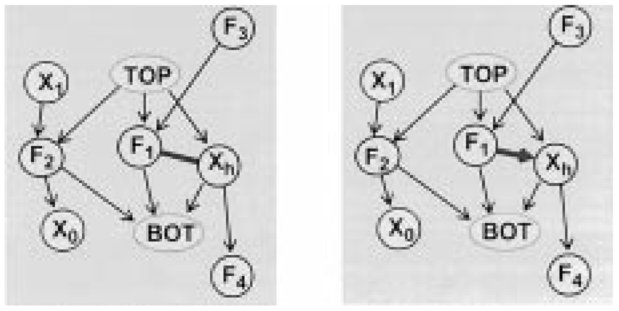


Fig. 7. Graph representation. (a) Before edge assignment $e(F_1, X_h)$ is undirected. (b) After edge assignment.

the top. DESW_i is the lower bound of node i's position if node i and its descendent chains are fully compacted toward the bottom. An overlapping happens from floating node i to its neighbors if $\mathrm{ANCW}_i < \mathrm{DESW}_i$. Cycles cannot occur in the graph because nodes associated with cycles cannot be implemented physically. For example, directed edges $d(n1, n2)$, $d(n2, n1)$ lead to ($\mathrm{ANCW}_{n1} > \mathrm{ANCW}_{n2}$), ($\mathrm{ANCW}_{n2} > \mathrm{ANCW}_{n1}$), respectively. Therefore, the cycle between $n1$ and $n2$ makes contradictory physical meaning.

The ANCW and DESW of fixed nodes are fixed. To check the possible overlapping between fixed nodes and other nodes, we define the derivative ancestor weight (DANCW) and the derivative descendent weight (DDESW) of fixed node X_n as follows:

1) $\mathrm{DANCW}_{Xn} = \min(\mathrm{ANCW}_i - wt(i, X_n)), d(i, X_n) \in E_{\mathrm{directed}}$;
2) $\mathrm{DDESW}_{Xn} = \max(\mathrm{DESW}_j + wt(X_n, j)), d(X_n, j) \in E_{\mathrm{directed}}$.

DANCW_{Xn} is essentially the lowest $\mathrm{pos}(X_n)$ bound to the longest directed path from T_c and DDESW_{Xn} is the upmost $\mathrm{pos}(X_n)$ bound to the longest directed path to B_c. For a fixed node X_n, it is a design rule error if $\mathrm{DANCW}_{Xn} < \mathrm{ANCW}_{Xn}$ or $\mathrm{DDESW}_{Xn} > \mathrm{DESW}_{Xn}$, where ANCW_{Xn} and DESW_{Xn} are given by (x or y). By using this constraint graph, we shall transform the crosspoint assignment problem into an FCRP in the next section.

C. FCRP

To construct a constraint graph for a layout, fixed nodes are first created for blockages. Directed edges and undirected edges are added between nodes to prevent design rule errors. The direction assignment on an undirected edge $e(i, j)$ determines the order between node i and node j. It also means the order of the crosspoints associated with node i and node j is determined. Fig. 7 shows an example in which the undirected edge $e(F_1, X_h)$ is assigned with a proper direction and therefore the node order between F_1 and X_h is determined. Since

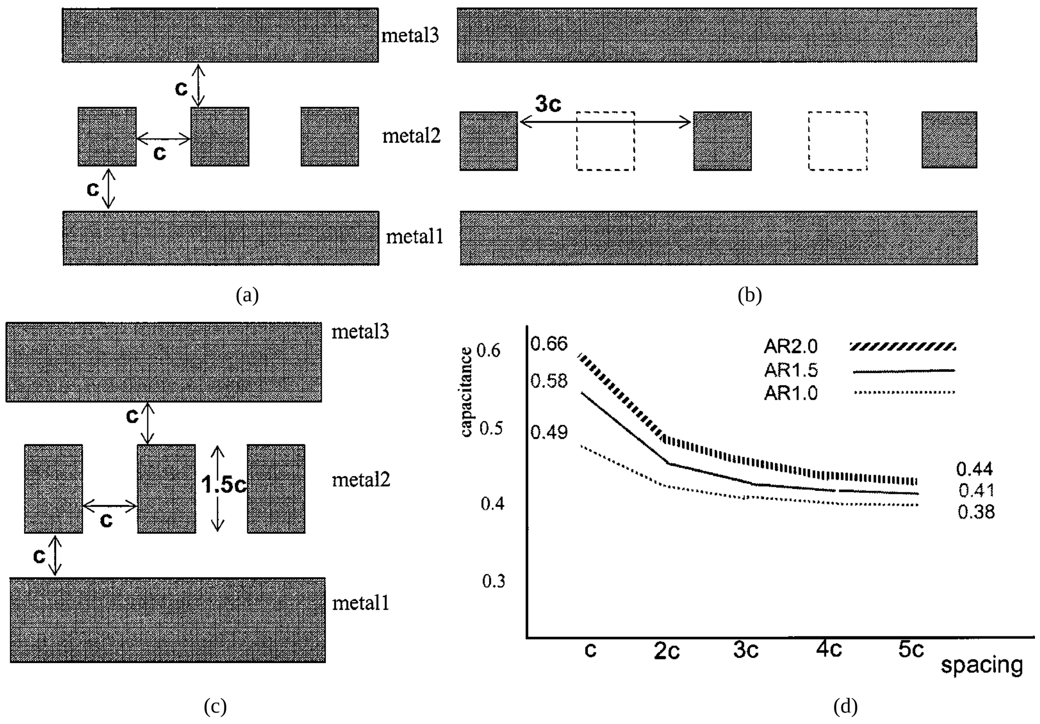


Fig. 8. Interconnect geometry and capacitance versus spacing ($\mathrm{SF} = 1$). (a) Minimum wire width with variable spacing. (b) Minimum wire width with one-track spacing. (c) Minimum spacing with wire aspect ratio = 1.5. (d) Capacitance versus spacing.

the area of a layout is fixed after the global routing stage, the goal of the optimization process is not to reduce the channel height but to improve timing and routability by determining net ordering and wire spacing. The height of the channel constraint graph (defined in [3]) is therefore fixed. We denote the new problem using a fixed-height channel constraint graph as the FCRP.

The main effort to enhance the routability of the solution of the FCRP is to straighten segments as much as possible. To optimize interconnect timing, we propose an efficient node order assignment algorithm in Section V and a space relaxation algorithm in Section VI. The timing optimization process measures the path delay on the fly and only targets the critical nets for delay reduction. We describe the capacitance extraction model and the delay calculation in the next section.

IV. Frame Work—*RC* Extraction Model and Interconnect Delay

A. *RC Extraction Model*

In our capacitance extraction model, we assume the layer above and the layer below a signal line are fully covered by metal, except the topmost metal layer which does not have an upper layer (see Fig. 8). Even if a layer above or below is only 50% covered with metal, the vertical capacitance (also commonly called "substrate capacitance") from a 50% covered metal layer is close to that from a fully covered metal layer due to the fringing effect. We ran a capacitance solver [1] to build up a lookup table for capacitance calculation using three parameters: parallel running length, distance, and wire aspect ratio. The wire geometry aspect ratio (height/width) was tested at 1.0, 1.5, and 2.0 for the lookup table. The effective capacitance of wire w is the summation of the contributions from the upper layer $C_{\mathrm{upper},\,w}$, the lower layer $C_{\mathrm{lower},\,w}$, and the intralayer neighboring wires $\sum(C_{j,\,w} * \mathrm{SF}_{j,\,w})$, where $\mathrm{SF}_{j,\,w}$ is the switching factor between two coupled signals through wires j and w. In Fig. 8, the distance to the neighboring wires is minimum spacing in (a) and increased to one-track spacing in (b). Fig. 8(c) shows the metal wires at geometric aspect ratio 1.5. Without considering signal switching activities (switching factor $\mathrm{SF} = 1$), the wire capacitance for various wire geometric aspect ratios (AR1.0, AR1.5, AR2.0) versus the spacing to neighboring wires is drawn in Fig. 8(d). By adding one blank track at each side of a wire, the wire capacitance is reduced by 23% for aspect ratio 1.0 and 33% for aspect ratio 2.0. As the figure indicates, the reduction in coupling capacitance due to extra spacing is very close to the minimum at $3c$ (one blank track). Therefore, in our crosspoint assignment algorithms, we at most add one extra track of spacing around sensitive critical wires. If the neighboring wires switch at the same time in different directions, we double the amount of intralayer coupling capacitance between neighbors (switching factor $\mathrm{SF} = 2$) for the delay calculation.

In Table I, the capacitances for wires at minimum distance and separated by one blank track when the switching factor is 1 and 2 are listed. For case 1, where the neighboring wires switch in different directions at the same time ($\mathrm{SF} = 2$) with minimum spacing, the intralayer capacitance is doubled from that for silent neighboring wires. By adding shielding (ground) wires in case 2 ($\mathrm{SF} = 1$), the coupling capacitance is reduced to be the same as that for silent neighboring wires. Clearly, in the presence of crosstalk ($\mathrm{SF} = 2$), adding one blank track around a sensitive wire (case 3) is more useful than adding shielding wires (case 2) in terms of delay. As Table I shows, adding one

TABLE I
COMPARISON OF COUPLING CAPACITANCE FOR VARIOUS SWITCHING FACTORS, ASPECT RATIOS, AND SPACINGS

	minimum spacing		separated by one blank track	
	case 2	case 1	case4	case3
	SF=1 (improv.% over case1)	SF=2	SF=1 (improv% over case2)	SF=2 (improv.% over case1)
AR1.0	0.49 (19%)	0.666	0.397(18%)	0.42 (32%)
AR1.5	0.58 (35%)	0.90	0.44 (24%)	0.48 (46%)
AR2.0	0.66 (40%)	1.10	0.467 (29%)	0.523 (52%)

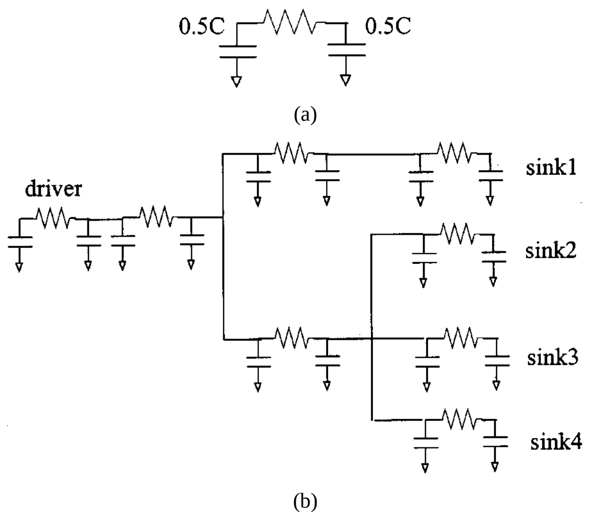


Fig. 9. (a) Wire segment and (b) distributed *RC* tree model.

blank track consistently gives better capacitance reduction than adding shielding wires.

In our framework, the position of a component in the layout is determined by the ANCW of the node that it corresponds to. The capacitance of a wire is interpolated from the values in the lookup table using geometric parameters (parallel-overlapping length, distance, wire width). The resistance of a wire is $l/\sigma w^2 AR$, where σ is conductivity, w is wire width, l is wire length, and AR is the wire geometric ratio. The path-delay calculation of distributed *RC* networks is described in the next section.

B. Interconnect Delay

The Elmore delay calculation [2] is used to estimate path delays. A model is used for each wire segment as shown in Fig. 9(a). The Elmore delay of path $P_p = \{s_{p0}, s_{p1}, \ldots, s_{pN-1} | s_{pi}$ is a segment on $P_p\}$ from driver to sink p is the summation of the delay contribution of all the components on the path. The delay contribution of a segment s_i is

$$\text{ED}_{Si} = \sum_j R_{Si} \bullet (0.5 C_{Si} + C_{Sj})$$

where Sj is a downstream segment, C_{Si} is the capacitance of s_i, and R_{Si} is the resistance of s_i. We obtain the Elmore path delay $\text{ED}_{Pp} = \sum_{s_i \in P_p} \text{ED}_{Si}$.

From the path-delay expression ED_{Pp}, we extract the delay contribution of one segment s_i in terms of capacitance as

$$\text{EDC}_{Si} = (C_{Si \bullet} \sum_k R_{Sk} + 0.5 C_{Si \bullet} R_{Si})$$

where $s_k \in P_p$ is a upstream segment. Our heuristics for the TDCA problem are mainly developed to reduce interconnect coupling capacitance rather than interconnect resistance. If jogs are not introduced, resistance should stay the same. Therefore, the segment having the largest Elmore delay contribution EDC among the interesting critical nets is processed with a higher priority.

V. NET ORDERING ALGORITHM

The difference between the FCRP and the conventional CRP is that the channel height in the FCRP is fixed and obstacles may appear in the middle of a channel. One way to guarantee the completion of crosspoint assignment for an FCRP is to allocate only one crosspoint for each node as shown in Fig. 10(a). This strategy usually generates a jog between two consecutive crosspoints and thus degrades detailed routability significantly and also increases wire length. One other more global way to tackle FCRP is that all of the crosspoints with the same signal are allocated into one node [as shown in Fig. 10(b)]. Then, all of the horizontal constraints and vertical constraints are added in the graph initially [as shown in Fig. 10(c)]. FCRP can therefore be solved using the Glitter channel router [3] by determining the direction assignment of undirected edges along with cycle breaking and jog insertion. However, we encounter the difficulty in jog insertion inside a constraint graph because a constraint graph does not provide information on wire density and routability on a G cell basis. As shown in Fig. 10(b), the straight segment has overlapping with three blockages and needs to be broken (by adding jogs) into smaller segments. This strategy has difficulty with maintaining obstacles in the graph and it tends to generate a large number of jogs at random locations under the fixed area constraint. It may end up degrading detailed routability.

As our experimental results will show in Section VII, the delay improvement by net ordering accounts for about one half of the total improvement, where relaxing spacing accounts for the other half. We implemented our net ordering algorithm such that all the pins on a selected boundary are assigned at once starting from the most congested boundary and proceeding outward. In Section V-A, we describe how to construct an initial graph by partitioning channels. The net ordering algorithm is presented in Section V-B and then the complexity analysis is given in Section V-C.

A. Constraint Graph Construction

Before we add any jogs, all segments with the same signal in a channel are straightened and represented by a single node [as in Fig. 10(b)]. A node n inside channel c initially has only

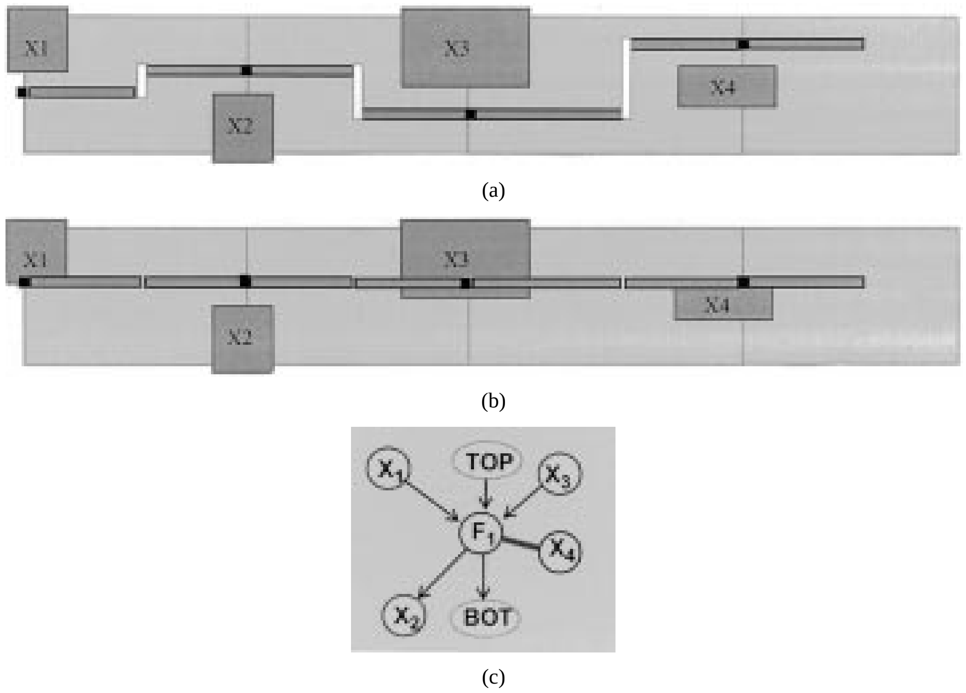


Fig. 10. Approaches to solve FCRP. (a) Each crosspoint is represented by a node. (b) All crosspoints with the same signal are represented by the same node. (c) Graph representation of (b).

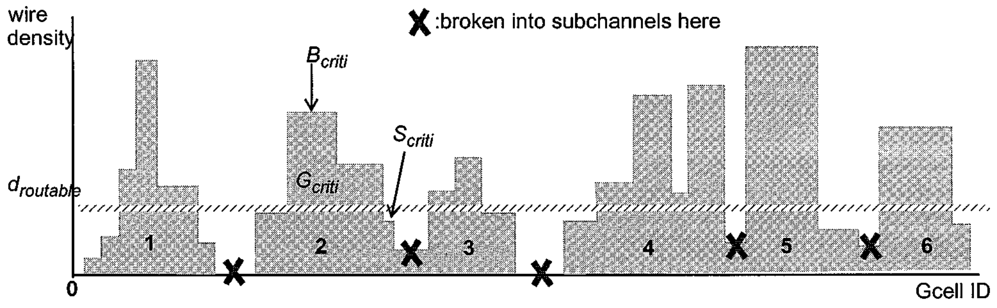


Fig. 11. Channel partitioning according to wire density.

one vertical constraint from top node T_c and one constraint to bottom node B_c. One technique to increase the solution space of our approach is to divide a channel into subchannels based on the wire density. In Fig. 11, the wire density along a channel may drop to below d_{routable}. At the density d_{routable}, provided by the user, signals should be easily routed successfully with jogs and doglegs. For example, $d_{\mathrm{routable}} = 2$ is reasonable for a 10 × 10 track G cell. By using a different node to represent each broken segment, we break straightened segments inside a G cell whose wire density is below d_{routable}. For example, the channel in Fig. 11 is broken into six groups (subchannels).

B. Algorithm

We calculate path delays on the fly during the optimization process. To speed up the performance, only certain interesting nets are selected for capacitance extraction and timing optimization. Based on information from synthesis, placement, and global routing, it is already known which nets potentially may exceed their timing budget. Initially we assume each node of the interesting nets has the worst possible neighboring wires and then we do capacitance extraction. A net having a negative timing slack value is called a *critical net* if its delay exceeds the user-specified timing requirement. Nodes on a critical net are called *critical nodes* and crosspoints for a critical node are called *critical crosspoints*.

The node n_{criti} in the timing critical delay path P having the largest Elmore Delay Contribution (EDC) among all the nodes on P is called the *most critical* node (segment) of P. Let S_{criti} be the most critical segment (node) of the most critical path P_c. The net ordering algorithm always processes the most critical subchannel G_{criti} that has the most critical segment S_{criti} (as shown in Fig. 11). It starts from the most congested boundary B_{criti} inside subchannel G_{criti} and proceeds outward. We show the basic flow of the net ordering algorithm in Fig. 12.

A crosspoint is *processed* if it is inserted into the graph. A node is processed if any one of its crosspoints is inserted into a graph. The crosspoints on a selected boundary are inserted into the graph in the following order:

1) the crosspoints whose nodes are fixed;
2) the crosspoints whose nodes have been processed, starting from bottommost one;
3) the rest of the crosspoints.

```
1: select the most critical path $P_c$, $timing\_slack_{Pc} = min(timing\_slack)$
2: select the most critical segment $S_{criti}$ on $P_c$
3: select the subchannel $G_{criti}$ {boundary: $b_0, \ldots, b_{B-1}$} which has $S_{criti}$
4: Let $Forward\_B = Backward\_B$ = the most congested boundary of $G_{criti}$
5: do{
6:     if ($Forward\_B <= b_{B-1}$) {
           do crosspoint assignment on boundary $Forward\_B$
           update timing slack of nodes on $Forward\_B$
       }
7:     if ($Forward\_B$ != $Backward\_B$) && ($Backward\_B >= b_0$){
           do crosspoint assignment on boundary $Backward\_B$
           update timing slack of nodes on $Backward\_B$
       }
8:     $Forward\_B$ ++; $Backward\_B$ --;
9: }while( $Forward\_B <= b_{B-1}$) || ($Backward\_B >= b_0$)
10: goto 1
```

Fig. 12. Basic flow of net ordering algorithm.

For each new crosspoint insertion, all the geometric intervals between the existing assigned crosspoints and obstacles are examined to satisfy design rules. The interval with the minimum cost is selected and the crosspoint is inserted into the graph. The cost function to assign crosspoint c_j in the interval (c_i, c_{i+1}) is equal to the coupling capacitance increase among critical nets in the system. Therefore, the objective to find a lowest cost crosspoint assignment is directly connected to a lowest coupling capacitance increase among critical nets. The cost function is described as follows:

$$\text{Cost} = \sum_{\text{after}} \text{Cap} - \sum_{\text{before}} \text{Cap}$$

$$\sum_{\text{before}} \text{Cap} = C_{\text{intra}}(i, i+1)(\delta_i + \delta_{i+1}) + C_{\text{rest}}$$

$$\sum_{\text{after}} \text{Cap} = C_{\text{intra}}(i, j)(\delta_i + \delta_j) + C_{\text{intra}}(j, i+1)(\delta_j + \delta_{i+1}) + \text{Inc\_}C_{\text{vert}} + C_{\text{rest}}$$

$$\text{Inc\_}C_{\text{vert}} = C_{\text{vert}}(j) + \sum_{d \in \text{critical descendents}(j)} \text{Inc}(C_{\text{vert}}(d))$$

where $\text{Inc\_}C_{\text{vert}}$ is the change in capacitance due to the change in lengths of the vertical segments connected to the two ends of the segments of interest. It includes the contribution from the current segment c_j and those from the critical descendents ($\text{timing\_slack} < 0$) of c_j. C_{rest} is the coupling capacitance of the rest of system except that in the interval (c_i, c_{i+1}). The critical measure $\delta_k = 1$ if crosspoint k is critical, otherwise $\delta_k = 0$.

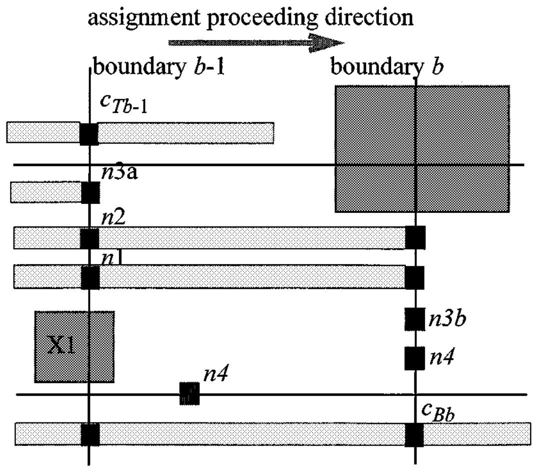
If all crosspoints of boundary b inside a region are assigned, we say boundary b is processed. For capacitance extraction purposes, C_{intra} of each unprocessed crosspoint $c_j \in s_n$ is calculated assuming that c_j has two worst neighbors on its boundary which gives the largest coupling capacitance.

Let the existing components (crosspoints or blockages) on boundary b be $c_1, \ldots, c_n$ and the bottommost component on boundary b of the upper channel be c_{Tb} and the topmost component on boundary b of the lower channel be c_{Bb} (see Fig. 13). Suppose crosspoint c_I is about to be assigned. Its node is n_I. The assignments between pairs of consecutive components $\{(c_{Bb}, c_1), \ldots, (c_n, c_{Tb})\}$ are all verified for design rules. An assignment of c_I on boundary b is valid if it satisfies $\text{ANCW}_{nI} > \text{DESW}_{nI}$. The maximal space interval between c_i and c_{i+1} is equal to $\text{ANCW}n_{ci} - \text{DESW}n_{ci+1} - 0.5\text{WIREWIDTH}c_i - 0.5\text{WIREWIDTH}c_{i+1}$. If none of these assignments is valid and there are intervals large enough to accommodate c_I, we break n_I between the previous boundary $b-1$ and the current boundary b. Then the new node of c_I is inserted into boundary b. If there is no interval large enough for c_I, we rip up all crosspoints on boundary b and break all nodes between boundary b and $b-1$. The new nodes of ripped crosspoints are assigned back to boundary b from the bottom in the ascending order of their old ANCW because we want to preserve the net ordering inherited from boundary $b-1$ for better routability.

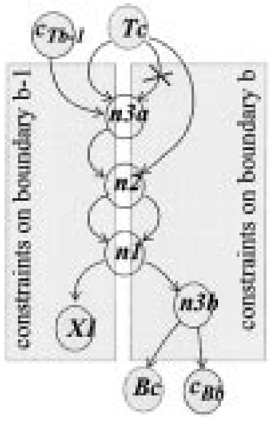
In Fig. 13, boundary $b-1$ is first processed and then it proceeds to boundary b. The crosspoint for net $n3$ on boundary b is tested and cannot be extended; therefore, we break it into two nodes: $n3a$ and $n3b$. Node $n3b$ is then inserted into boundary b at the minimum cost. Node $n4$ has a crosspoint at the bottom of the Gcell and one at boundary b. Since the crosspoints of node $n4$ are not processed, the crosspoint of $n4$ on boundary b is inserted after $n3b$ is inserted.

C. Time Complexity of Net Ordering Algorithm

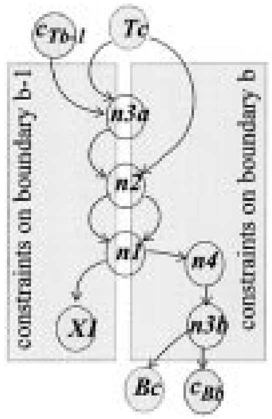
The net ordering algorithm has two major computational parts—boundary pin assignment and delay calculation. The dimension of the G cell plane is $O(G \bullet G)$ and the size of a G cell is $O(T \bullet T)$ in units of tracks. The computation time to assign pins on a single boundary is $O(T!)$. There are $T \bullet p$ critical nets passing through a boundary, where p is the percentage of nets that are timing critical. The average number of segments of a net is $O(G)$, therefore the time to calculate the Elmore delay of a critical net is $O(G)$. After each boundary is processed, the Elmore delay of the critical nets through that boundary is calculated in $O(G \bullet T \bullet p)$ time. The total time complexity to do pin assignment and delay calculation for the whole circuit is $O(G^2(G \bullet T \bullet p + T!))$. In practical industrial



(a)



(b)



(c)

Fig. 13. Crosspoint assignment and its graph representation. (a) Boundary b is processed after boundary $b-1$. (b) $n3$ is broken to $n3a$ and $n3b$. (c) Insert $n4$.

circuits, the size of a G cell is fixed (e.g., 7 × 7 to 20 × 20 tracks) and not scalable. Therefore, the time complexity is $O(G^3Tp)$.

VI. Space Relaxation Algorithm

In practical circuits, not all of the G cells are fully congested. We aim to utilize the remaining routing resources in the uncongested G cells to reduce the coupling capacitance of critical nets. Based on the fixed channel graph in which net ordering has been assigned, we further increase the edge weights of critical nodes to push neighboring nodes away. Two techniques are proposed to relax spacing in Section VI-A.

A. Algorithm

In a channel graph, there may be multiple critical nodes to which we need to allocate routing resources. Since there may be multiple directed paths connecting two critical nodes in a channel graph, the space allocation for one node may degrade the possible space allocation of other critical nodes on the same path. To alleviate the ordering problem in space allocation problems, we allocate only partial space resources around a node at each iteration. At each iteration, the *critical* node n_c with the largest Elmore delay contribution EDC among the most critical net N_c $(\mathrm{timing\_slack}_{N_c} = \min(\mathrm{timing\_slack}))$ is selected for space relaxation. The spatial slack of node n_c is equal to $(\mathrm{ANCW}n_c - \mathrm{DESW}n_c)$, which is analogous to the geometric movable range of node n_c. Two stages of relaxation optimization are applied to the most critical node at each iteration. The flow of the space relaxation algorithm is in Fig. 14.

In line 1 of Fig. 14, there are two types of optimization objectives. $\mathrm{Obj1}$ is to select all the timing critical paths and add spacing to their most timing critical nodes iteratively. $\mathrm{Obj2}$ is to only select the most timing critical path and add spacing to its most critical node. $\mathrm{Obj1}$ reduces capacitance of all the critical nets, but $\mathrm{obj2}$ only reduces that of the net with the worst timing. In our experiments, we only run $\mathrm{obj1}$ and show the overall delay reduction in all the timing critical nets. The self relaxation and neighbor relaxation algorithms are described in the following paragraphs.

Node i is said to be *self-relaxable* if $\mathrm{ANCW}i > \mathrm{DESW}i$, i.e., $\mathrm{spatial\_slack}(i) > 0$. The movable range of node i is $[\mathrm{ANCW}i, \mathrm{DESW}i]$. That is, the position of node i can be anywhere between $\mathrm{ANCW}i$ and $\mathrm{DESW}i$. At every iteration, the most critical node n_c is selected to either add space to all neighbors on one side of itself (*self relaxation*) or add space to only one selected neighbor (*neighbor relaxation*). If n_c is self-relaxable, we need to decide which one of the upper and lower sides of n_c should have space added based on the least cost. The cost function for n_c's self-relaxation for adding spacing to all the *adjacent ancestors* $AAnc(n_c)$ is as follows:

$$\begin{aligned}
&\mathrm{Cost}_{\mathrm{self\_relax\_anc}} \\
&\quad = \sum_{\mathrm{after}} \mathrm{Cap}(n_c,\, AAnc(n_c),\, \mathrm{dist}_{\mathrm{after}}) \\
&\qquad - \sum_{\mathrm{before}} \mathrm{Cap}(n_c,\, AAnc(n_c),\, \mathrm{dist}_{\mathrm{before}}) + \mathrm{Inc}(C_{\mathrm{vert}}(n_c)) \\
&\qquad + \sum_{d \in \mathrm{critical\ descendents}(n_c)} \mathrm{Inc}(C_{\mathrm{vert}}(d))
\end{aligned}$$

where $\mathrm{Inc}(C_{\mathrm{vert}}(n_c))$ is the capacitance increase from the vertical trunks connected to node n_c. The self relaxation cost

```
1: select timing critical path $p$ and add it to $P_c$, where $p$ is not disabled and
        ($Obj1$: $timing\_slack_P < 0$; $Obj2$: $timing\_slack_P = min(timing\_slack)$)
2: $LIST$: list of nodes to be given extra spacing
3: for each critical path $p \in P_c$ {
4:      select $n_c \in p$, where $n_c$ is relaxable, $EDCn_c = max(EDC_n)$, $n \in p$
5:      if ($n_c = \Phi$){disable $p$}
6:      else{insert $n_c$ into $LIST$;}
7: }
8: while ( $n \in LIST$){
        node $n$ has neighbors { $nn_0, ..., nn_{N-1}$}
9:      if ($spatial\_slack_n > 0$)        { do self-relaxation on node $n$}
10:     else if (at least one neighbor $nn$ has $spatial\_slack_{nn} > 0$) { do neighbor-relaxation on neighbors}
11:     else{ delete $n$ from $LIST$ }
12: }
13: update the timing slack values of moved nodes
14: if there is at least one timing critical path $p$ that is not disabled, then goto 1
```

Fig. 14. Pseudocode of space relaxation.

function for adding spacing to all the *adjacent descendents* $\mathrm{ADes}(n_c)$ is as follows:

$$\begin{aligned}\mathrm{Cost}_{\mathrm{self\_relax\_des}} &= \sum_{\mathrm{after}} \mathrm{Cap}(n_c,\, \mathrm{ADes}(n_c),\, \mathrm{dist}_{\mathrm{after}}) \\ &\quad - \sum_{\mathrm{before}} \mathrm{Cap}(n_c,\, \mathrm{ADes}(n_c),\, \mathrm{dist}_{\mathrm{before}}) \\ &\quad + \sum_{d \in \mathrm{critical\ descendents}(n_c)} \mathrm{Inc}(C_{\mathrm{vert}}(d)).\end{aligned}$$

If $C_{\mathrm{self\_relax\_anc}} > C_{\mathrm{self\_relax\_des}}$, we add an extra spacing $\mathrm{Extra\_Space} = \min(\mathrm{Increment\_Space},\ \mathrm{ANCW}n_c - \mathrm{DESW}n_c)$ above n_c by increasing $wt(\mathrm{AAnc}(n_c),\, n_c)$ to $\max(wt(\mathrm{AAnc}(n_c),\, n_c),\ \mathrm{Upper\_Min\_Space} + \mathrm{Extra\_Space})$, where $\mathrm{Upper\_Min\_Space}$ is the minimum space from n_c to all adjacent ancestors and $\mathrm{Increment\_Space}$ is a user-specified number (e.g., 0.5 track). If $C_{\mathrm{self\_relax\_anc}} < C_{\mathrm{self\_relax\_des}}$, we add an extra spacing $\mathrm{Extra\_Space} = \min(\mathrm{Increment\_Space},\ \mathrm{ANCW}n_c - \mathrm{DESW}n_c)$ below n_c by increasing $wt(\mathrm{ADes}(n_c),\, n_c)$ to $\max(wt(\mathrm{ADes}(n_c),\, n_c), \mathrm{Lower\_Min\_Space} + \mathrm{Extra\_Space})$, where $\mathrm{Lower\_Min\_Space}$ is the minimum space from n_c to all adjacent descendents. The self-relaxation is continued until either $\mathrm{spatial\_slack}(n_c) = 0$ or the minimum spacing to n_c's neighbors exceeds one-blank-track spacing. Then, it proceeds to the second stage of neighbor relaxation.

If n_c is not self-relaxable, its relaxable neighbor with the lowest cost is selected for neighbor relaxation at each iteration. The cost to add spacing between n_c and its movable neighbor n_i is as follows:

$$\begin{aligned}\mathrm{Cost}_{nei\_\mathrm{relax}} &= C_{\mathrm{intra}}(n_c,\, n_i,\, \mathrm{dist}_{\mathrm{after}}) \\ &\quad - C_{\mathrm{intra}}(n_c,\, n_i,\, \mathrm{dist}_{\mathrm{before}}) \\ &\quad + \mathrm{Inc}(C_{\mathrm{vert}}(LW)) \\ &\quad + \sum_{d \in \mathrm{critical\ descendents}(n_c)} \mathrm{Inc}(C_{\mathrm{vert}}(d))\end{aligned}$$

where LW is the one of n_c and n_i whichever has the least ANCW.

Critical neighbors are processed prior to noncritical neighbors. To add extra spacing between n_c and selected neighbor n_i, the edge weight $wt(n_i,\, n_c)$ is incremented by $\min(\mathrm{Increment\_Space},\ \mathrm{ANCWn_i} - \mathrm{DESWn_i})$. The neighbor relaxation process is continued until either $wt(n_i,\, n_c)$ exceeds one-blank-track spacing or $\mathrm{spatial\_slack}(n_i) = 0$. Note that the two relaxation stages (self-relaxation and neighbor relaxation) add spacing also to neighboring nodes outside of the local channel by increasing the weights of the edges connected the neighbors outside of the local channel. The spacing between two nodes inside two adjacent channels is relaxed by adding edge weight between the two. We show an example of self-relaxation and neighbor relaxation on node 4 in Fig. 15. The light shaded region of each node in Fig. 15(a) stands for the movable region and the dark nodes in Fig. 15(b) are nonrelaxable because extra spaces (dark shaded areas) are added and its spatial slack $= 0$. The relaxable neighbors $n1$, $n3$, $n5$, $n6$ are further relaxed outward by adding extra edge weight until the spatial slack $= 0$.

B. Time Complexity of Spacing Algorithm

We first calculate the runtime for space relaxation on a critical segment. The dimension of the G cell plane is $G \bullet G$. The size of a G cell is $T \bullet T$ in tracks. The average number of segments on a net is $O(G)$. There are C critical nets, so $O(CG)$ segments need to be space relaxed. Self-relaxation for each node takes at most four iterations and neighbor relaxation takes at most two iterations for each pair of nodes, if $\mathrm{Increment\_Space} = 0.5$ track. In the worst case, $O(GT)$ nodes need to have their ANCW and DESW recalculated if extra spacing is added between a pair of nodes. Therefore the execution time for the self relaxation of a node is $O(GT)$. A node has at most $2G$ neighbors, thus the runtime for the neighbor relaxation of a node is $O(G \bullet GT)$. The execution time for space relaxation on a node becomes $O(G^2T)$. The total runtime to relax spacing of critical segments is $O(CG^3T)$.

Secondly, we want to calculate the runtime for Elmore delay calculation. The average number of segments of a net is G,

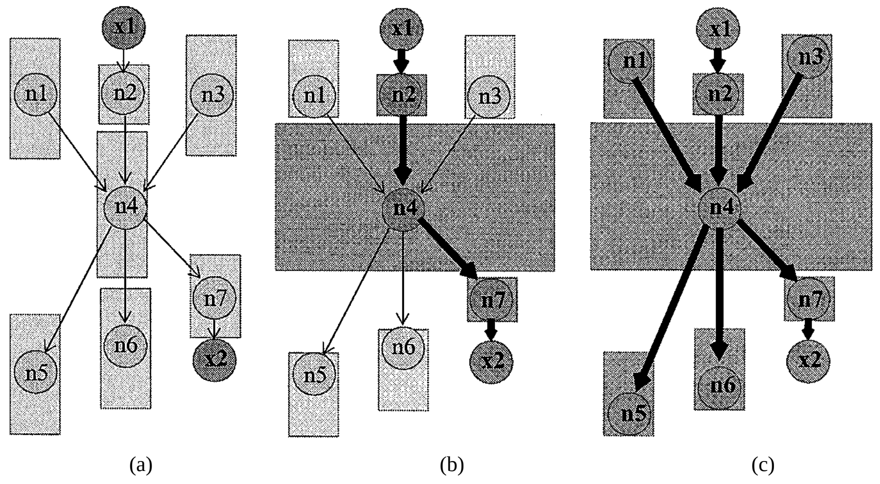


Fig. 15. (a) Movable ranges (light shaded). (b) Self-relaxation of $n4$. Nodes with zero spatial slack are shaded. (c) Neighbor relaxation of $n1$, $n3$, $n5$, and $n6$.

TABLE II
CHARACTERISTICS OF BENCHMARK CIRCUITS

number of	cells	nets	pins	cross points	No. of global cells	minimum area (E6 mfs^2)	
						2 metal layer	4 metal layer
hp	11	83	309	1375	19 X 25	13	12.3
ami33	37	124	513	2611	35 X 42	2.44	1.89
apte	9	96	283	938	18 X 20	54.3	49.6
xerox	10	203	696	2132	19 X 23	26.8	24.6
ami49	49	408	953	13360	88 X 96	48.2	46.4

therefore the time to calculate the Elmore delay of a critical net is G. Each step of space relaxation changes at most the timing delay of GT nodes in the same channel. In the worst case, the timing delay of moved nodes needs to be calculated at each step of space relaxation. The maximum number of space relaxation steps is (self relaxations + neighbor relaxations) = $(2 \bullet CG + 4 \bullet CG \bullet 2G) = O(CG^2)$. Therefore the time complexity to update timing slack database is $O(G \bullet GT \bullet CG^2) = O(CG^4T)$.

The overall time complexity of the spacing algorithm becomes $O(CG^4T)$.

VII. Experimental Results

We have implemented the timing- and crosstalk-driven prerouter in GNU C++ on Unix systems. To test the MCNC macrocell benchmark circuits, we generate placements by the TimberWolfMC placer [18], global routing by a Steiner tree heuristic [19], and complete detailed routing by the Kokanee tile-based router [20].

The characteristics of the tested MCNC benchmark circuits are listed in Table II. Area and length in tables are in the unit of minimum feature size (*mfs*) of its fabrication process. In all of our tested circuits, we use *mfs* $= 0.25$ μm.

The two-layer layouts for the tested circuits are shown in Fig. 16. In Table III, we select the six largest nets from *ami33*, each of which has more than 28 pins, as critical nets (actual timing information is not available for these benchmarks). For each large net n_c in each column, the switching factor $\mathrm{SF}_{nc,i} = 2$ and $\mathrm{SF}_{i,j} = 1$, where $i \neq j$, $i \neq n_c$, $j \neq n_c$. In the rightmost column, we let all of the selected large nets be critical nets $\{n_{c1}, \ldots, n_{c6}\}$ and optimize their interconnect while setting switching factor SF $= 2$, where $i \neq j$, and the rest SF $= 1$. We obtain a 5%–16% path-delay reduction for wire geometric aspect ratio 1.0, an 8.2%–21% reduction for aspect ratio 1.5, and a 9.7%–25% reduction for aspect ratio 2.0. The path-delay reduction varies quite differently from net to net because it highly depends on the wire density of the regions where a net goes into. In the rightmost column of Table III, "$N + S$" stands for applying the net ordering and spacing optimizations and "N" means only the net-ordering optimization is applied. It shows that the net-ordering algorithm accounts for about 45% of the timing improvement.

Elmore delay can be less accurate if fanout is large. To give a more accurate estimation of delay reduction, we will only show the optimization results on two-pin nets in the following tables. In Table IV, we select six groups of two-pin nets and these six groups are independent of each other. We obtain a 5%–13% delay reduction at aspect ratio 1.0, a 6.8%–15.4% reduction at aspect ratio 1.5, and a 8.4%–18.5% reduction at aspect ratio 2.0.

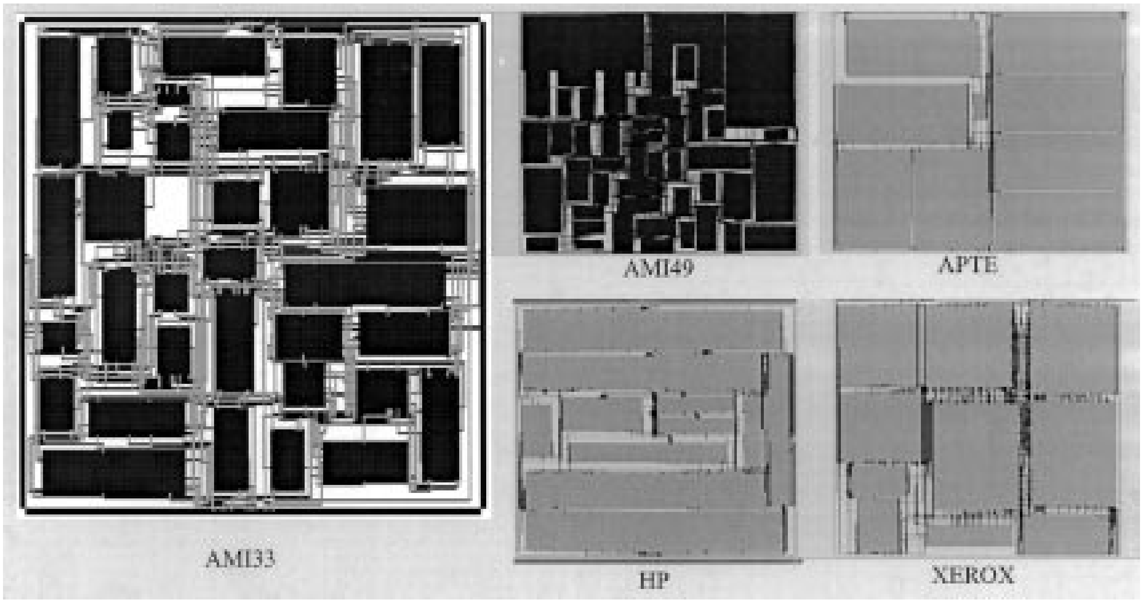


Fig. 16. Two-layer layout of tested MCNC benchmark circuits.

TABLE III
DELAY REDUCTION FOR LARGE PIN-COUNT NETS IN MCNC BENCHMARK ami33 IN TWO METAL LAYERS

net name		P1G	P1F	P2G	P_1	P_0	P2F	ALL	
pin counts (percentage among the whole circuit)		50 (10%)	44 (9%)	56 (11%)	29 (6%)	30 (6%)	47 (9%)	256 (50%)	
wire length (mfs unit)		11173	10120	11509	8969	9215	10232	61218	
								N+S	N
path delay reduction	AR1.0	-7%	-16%	-8%	-13.7%	-10.3%	-10%	-5%	-2.3%
	AR1.5	-11.4%	-21%	-13%	-16.8%	-14.9%	-14.3%	-8.2%	-3.6%
	AR2.0	-14.6%	-25%	-17%	-19.9%	-17.7%	-17.5%	-9.7%	-4%

TABLE IV
OPTIMIZATION FOR TWO-PIN AND THREE-PIN NETS IN ami33

group of 2pin nets	**1** (7 nets)	**2** (15 nets)	**3** (14 nets)	**4** (10 nets)	**5**(17 nets)	**6** (24 nets)	
wire length (% among the whole circuit)	1.7%	6.2%	5.8%	6.5%	3.4%	10.1%	
						N+S	N
AR1.0	-5%	-7%	-13%	-6%	-7%	-10%	-7%
AR1.5	-6.8%	-8.8%	-15.4%	-7.7%	-9.9%	-11%	-8%
AR2.0	-8.4%	-10.7%	-18.5%	-9.5%	-11.8%	-13.9%	-10.5%

In the rightmost column of Table IV, the net-ordering algorithm accounts for about 70% of the total improvement.

Some of the global routing cells are comparatively small and do not have enough routing resources to complete all the routes. The tile-based area router merges the failing global cells with neighboring global cells and completes the final routing in the increased areas. In Table V, for each column we optimize the timing of the selected groups from Table IV. The routability of pin assignment is measured by the number of failing nets before merging global cells, as shown in Table V. If we delete 5% of the net lengths ($0.95N$ in the table), this increases the amount of routing space in the G cells. It is done by randomly selecting nets for deletion until the total net length is reduced by 5%. As the table indicates and as we expected, the delay reduction increases in this case. Note that the routability is quite good in both cases, in fact, the detailed router has no difficulty in completing all connections after merging each (of the few) failed G cells with its neighboring Gcells.

TABLE V
PATH DELAY IMPROVEMENT AND ROUTING FAILURES VERSUS WIRE DENSITY IN ami33

# of critical nets (% of wire length in the circuit)		group 1 (1.7%)		group 5 (3.4%)		group 2&3 (12%)			group 4&6 (16.6%)		
		delay improv.	failed Gcells	delay	failed Gcells	delay		failed Gcells	delay		failed Gcells
						N+S	N		N+S	N	
AR1.0	1.00N	-5%	0.73%	-7%	0.83%	-10%	-7%	0.77%	-8.2%	-6%	0.77%
	0.95N	-6.3%	0.8%	-7.1%	0.8%	-11.8%	-7%	0.77%	-8.9%	-7.1%	0.83%

TABLE VI
PATH-DELAY IMPROVEMENT FOR TWO METAL LAYERS IN PERCENTAGE (10% NETS ARE CRITICAL)

circuit	ami33				ami49				hp				xerox				apte			
wire density	1.0	0.95	0.90	0.80	1.0	0.95	0.90	0.80	1.0	0.95	0.90	0.80	1.0	0.95	0.90	0.80	1.0	0.95	0.90	0.80
AR1.0	-9.5	-9.5	-11	-13	-9.2	-9.6	-10	-10.8	-7.3	-7.9	-8.7	-8.9	-11.4	-12.1	-12.2	-13.8	-9.7	-9.6	-12.6	-12.8
AR1.5	-11	-11	-11.5	-12.3	-13.6	-13.8	-14.5	-15.3	-10.8	-10.7	-12.7	-14.2	-13.9	-14.8	-16	-18.5	-13.7	-13.4	-11.8	-12.5
AR2.0	-13.9	-14.1	-13.2	-14	-15.9	-16.3	-18.8	-18.7	-12.5	-12.9	-15.2	-17	-15.8	-16.8	-18.3	-22	-16.4	-16	-14.2	-14.8

TABLE VII
PATH-DELAY IMPROVEMENT FOR FOUR METAL LAYERS IN PERCENTAGE (10% OF THE NETS ARE CRITICAL)

circuit	ami33	ami49	hp	xerox	apte
AR1.0	-5	-4.7	-1.3	-10	-0.6
AR1.5	-5.3	-6.4	-2.7	-13.3	-1
AR2.0	-9.2	-7.8	-4.4	-14	-1.4

In Table VI, for each circuit, we randomly chose a set of two-pin nets which accounts for 10% of the total wire length in each tested circuit. The selected two-pin nets are treated as critical nets while setting $\mathrm{SF} = 2$ between them and $\mathrm{SF} = 1$ otherwise. In Table VI, we delete 5%, 10%, and 20% of the net lengths to measure the impact on the timing improvement as more routing spaces are available, compared with the original circuits. They are denoted as 1.0, 0.95, 0.90, and 0.80 for 0%, 5%, 10%, and 20% net length deletion, respectively. The cases for two and four metal layers are tested in Table VI and Table VII, respectively. All of the tested circuits show a consistently increasing reduction in delay for lowering wire density and larger wire geometric ratio except the apte circuit. The apte circuit already has sparse wire density in most of the G cells and, therefore, little improvement is seen as we decrease the wire density. This is also the case for apte for the four-layer case, as shown in Table VII.

The data in Table VI is plotted in Fig. 17 for two metal layers. It illustrates the average performance improvement on these test cases by tuning net ordering and adjusting wire spacing without increasing chip area. The rightmost data point of each curve in the picture represents the case where C_{intra} is set to zero, shown as a theoretical lower bound for path-delay reduction. Our timing optimizer achieves 46% of the optimal path-delay reduction, on average, where the optimal delay reduction is obtained by putting infinite space around each wire segment. As we expect, the taller the wires are in a deep submicrometer process, the more path delay and C_{intra} reduction we can obtain.

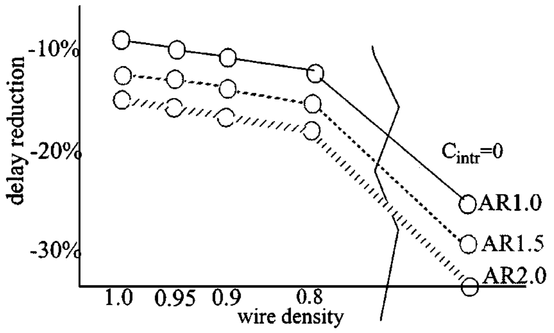


Fig. 17. Average path-delay reduction versus wire density in Table VI.

TABLE VIII
PATH DELAY IMPROVEMENT OF ALL NETS FOR TWO METAL LAYERS IN PERCENTAGE

circuit	ami33	ami49	hp	xerox	apte
AR1.0	-4.9	-5.0	-6.9	-9.1	-7.7
AR1.5	-7.1	-6.7	-9.7	-13.8	-11
AR2.0	-8.8	-8.4	-11.5	-16.2	-14

In Table VIII, we select all the nets for optimization and set every net as aggressor ($\mathrm{SF} = 2$) to each other. The value of our new method is clearly shown by a consistent 4%–16% delay reduction by our efficient heuristics at no routing area penalty. Please note that the placement and global routing results of all test cases are from [21]. They are the best reported core sizes to date. Our new approach demonstrates that it is possible to make significant improvement on coupling capacitance and still complete the routing of dense designs.

VIII. Conclusion

We have presented a timing- and crosstalk-driven router for the chip assembly task that is applied between global and detailed routing. Our new approach aims to process the crosstalk and timing constraints by ordering nets and tuning wire spacing in a quantitative way. The new approach fits between global routing and detailed routing along the physical design flow. It is the first to address the timing- and crosstalk-driven area routing problem using crosspoint assignment prior to the detailed routing stage, in contrast to the previous approaches which are applied in the post-detailed routing stage. Our new approach enjoys a larger optimization solution space than the previous approaches whose solution space is highly limited by routed geometric constraints. Based on the global routing information, our graph-based optimizer preroutes wires on the global routing grids incrementally. The graph-based optimizer has two stages, net order assignment and space relaxation. A quick capacitance extraction and Elmore delay calculator considering signal switching activities were implemented to find the timing of critical nets and to provide the timing slack database of critical nets. As the graph-based algorithm proceeds, the path delay of critical nets and the timing slack database are updated. During the optimization process, it only optimizes the timing critical paths with negative slack values. The experimental results showed a 5%–16% delay reduction for MCNC macrocell benchmark circuits for a 0.25-μm process for wire geometric ratio (height/width) $= 1.0$, against a 25% delay reduction if there is infinite space around each metal wire on the same layer. It showed a remarkable 8.4%–25% delay reduction for MCNC benchmarks for wire geometric ratio $= 2.0$, against a 33% delay reduction if there is infinite space around each metal wire. As experimental results indicated, our new approach overall achieved an average 46% reduction on intralayer coupling capacitance. Therefore, the delay reduction by our optimizer is more significant when the minimum feature size continuously shrinks and the wire geometric ratio increases.

Acknowledgment

The authors would like to thank Prof. A. Kahng of the University of California at Los Angeles for his helpful opinions on interconnection delay analysis.

References

[1] "bem2d 2D field capacitance solver," Cadence Design Systems, Inc, San Jose, CA.
[2] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, Jan. 1948.
[3] H. Chen and E. S. Kuh, "Glitter: A gridless variable-width channel router," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 459–465, Oct. 1986.
[4] W. C. Kao and T. M. Parng, "Cross point assignment with global rerouting for general-architecture designs," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 337–348, Mar. 1995.
[5] D. A. Kirkpatrick and A. L. Sangiovanni-Vincentelli, "Digital sensitivity: Predicting signal interaction using functional analysis," in *Dig. Tech. Papers IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1996, pp. 536–541.
[6] T. Xue, E. S. Kuh, and D. Wang, "Post global routing crosstalk risk estimation and reduction," in *Dig. Tech. Papers IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1996, pp. 302–309.
[7] T. Hameenanttila, J. D. Carothers, and D. Li, "Fast coupled noise estimation for crosstalk avoidance in the MCG multichip module autorouter," *IEEE Trans. Computer-Aided Design.*, vol. 4, pp. 356–368, Sept. 1996.
[8] H. Zhou and D. F. Wong, "An optimal algorithm for river routing with crosstalk constraints," in *Dig. Tech. Papers IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1996, pp. 310–315.
[9] D. A. Kirkpatrick and A. L. Sangiovanni-Vicentelli, "Techniques for crosstalk avoidance in the physical design of high-performance digital systems," in *Dig. Tech. Papers IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 616–619.
[10] A. Vittal and M. Marek-Sadowska, "Crosstalk reduction forVLSI," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 290–298, Mar. 1997.
[11] T. Gao and C. L. Liu, "Minimum crosstalk channel routing," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 465–474, May 1996.
[12] K. Jhang, S. Ha, and C. S. Jhon, "COP: A crosstalk optimizer for gridded channel routing," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 424–429, Apr. 1996.
[13] T. Gao and C. L. Liu, "Minimum crosstalk switchbox routing," in *Dig. Tech. Papers IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 610–615.
[14] E. Malavasi, U. Choudhury, and A. Sangiovanni-Vincentelli, "A routing methodology for analog integrated circuits," in *Dig. Tech. Papers IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1990, pp. 202–205.
[15] P. F. Dubois and L. O. Donzelle, "New advances in the routing of mixed analog and digital channels," in *Proc. IEEE Int. Symp. Circuits Systems*, vol. 4, June 1991, pp. 1956–1959.
[16] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint-based channel routing for analog and mixed analog/digital circuits," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 497–510, Apr. 1993.
[17] K. Chaudhary, A. Onozawa, and E. S. Kuh, "A spacing algorithm for performance enhancement and cross-talk reduction," in *Dig. Tech. Papers IEEE/ACM Int. Conf Computer-Aided Design*, Nov. 1993, pp. 697–702.
[18] W. Swartz and C. Sechen, "New alogrithms for the placement and routing of macrocells," in *Dig. Tech. Papers IEEE/ACM Int. Conf. Computer-Aided Design*, Nov. 1990, pp. 336–339.
[19] L. C. E. Liu and C. Sechen, "Multi-layer chip-level global routing using an efficient graph-based steiner tree heuristic," in *Proc. Eur. Design and Test Conf.*, Mar. 1997, pp. 311–318.
[20] H. P. Tseng, "Detailed routing algorithms for VLSI circuits," Ph.D. dissertation, Univ. Washington, Seattle, WA, 1997.
[21] L. C. E. Liu, H. P. Tseng, and C. Sechen, "Chip-level area routing," in *Proc. Int. Symp. Physical Design*, Apr. 1998, pp. 197–204.
[22] M. R. Garey and D. S. Johnson, *Computers and Intractability—A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.

Hsiao-Ping Tseng (S'95–M'98) received the B.S. degree in physics from the National Taiwan University, Taiwan, R.O.C., in 1991 and the M.S. and Ph.D. degrees in electrical engineering from the University of Washington, Seattle, in 1995 and 1997, respectively.

From 1998 to 1998, he was with Cadence Design Systems as a Physical Design Developer. Since May 1998, he has been a Senior Researcher at Magma Design Automation, Cupertino, CA. His research interests include signal integrity, interconnect analysis, and physical design automation for high speed circuits.

Louis Scheffer (SM'95) received the B.S. and M.S. degrees from the California Institute of Technology, Pasadena, in 1974 and 1975, respectively, and the Ph.D. degree from Stanford University, Stanford, CA, in 1984.

He was a Chip Designer and Computer-Aided Design Tool Developer with Hewlett Packard from 1975 to 1981. In 1981, he joined Valid Logic Systems and was engaged in hardware design, developing schematic editors, and building integrated-circuit layouts, routing, and system verification . He has been with Cadence since 1991, working on place and route and floorplanning systems. He enjoys teaching and has taught courses on computer-aided design for electronics at University of California, Berkeley, and Stanford University and serves on the technical advisory board for the Allen Telescope Array at the SETI institute. His main research interests include floorplanning and deep submicrometer effects.

Carl Sechen (S'74–M'85) received the B.E.E. degree from the University of Minnesota, Minneapolis, St. Paul, in 1977, the M.S. degree from the Massachusetts Institute of Technology, Cambridge, in 1979, and the Ph.D. degree from the University of California, Berkeley, in 1986.

From 1986 to 1992, he was an Assistant and then an Associate Professor at Yale University, New Haven, CT. Since July 1992, he has been an Associate Professor and now Professor in the Department of Electrical Engineering at the University of Washington, Seattle. His primary research interests are the design and computer-aided design of high-speed digital integrated circuits. He was the initial developer of the TimberWolf placement and routing package. He cofounded InternetCAD.com, Inc., a placement and routing software vendor.

Dr. Sechen received the Semiconductor Research Corporation's 1994 Technical Excellence Award and the Semiconductor Research Corporation's 1988 Inventor's Award.