Pattern Routing: Use and Theory for Increasing Predictability and Avoiding Coupling

Ryan Kastner, *Student Member, IEEE*, Elaheh Bozorgzadeh, *Student Member, IEEE*, and Majid Sarrafzadeh, *Fellow, IEEE*

***Abstract*—Deep submicron effects, along with increasing interconnect densities, have increased the complexity of the routing problem. Whereas previously we could focus on minimizing wirelength, we must now consider a variety of objectives during routing. For example, an increased amount of timing restrictions means that we must minimize interconnect delay. But, interconnect delay is no longer simply related to wirelength. Coupling capacitance has become a dominant component of delay due to the shrinking of device sizes. Regardless, the most important objective is producing a routable circuit. Unfortunately, this often conflicts with minimizing interconnect delay as minimum delay routes create congested areas, for which an exact routing cannot be realized without violating design rules. In this work, we use the concept of pattern routing to develop algorithms that guide the router to a solution that minimizes interconnect delay—by considering both coupling and wirelength—without damaging the routability of the circuit. The paper is divided into two parts. The first part demonstrates that pattern routing can be used without affecting the routability of the circuit. We propose two schemes to choose a set of nets to pattern route. Using these schemes, we show that the routability is not hindered. The second part builds on the previous part by presenting a framework for coupling reduction using pattern routing. We develop theory and algorithms relating pattern routing and coupling. Additionally, we give suggestions on how to extend our theory and use our algorithms for both global and detailed routing.**

***Index Terms*—Congestion, coupling, detailed routing, global routing, interconnect, pattern routing, physical design, routing estimation.**

I. Introduction

THE PROCESS of routing can be divided into two subproblems, global and detailed routing. Global routing decomposes the routing problem into smaller manageable routings for the detailed router. Specifically, the global router finds a rough path for each net while trying to reduce the chip size, decrease the interconnect delay, and distribute the congestion across the routing area, among other things [1]–[3]. Detailed routing uses the results of global routing to find an exact realization of the interconnections in VLSI circuits.

The global routing problem is known to be NP-hard [4]. This motivates the use of heuristic and approximation algorithms. The maze routing (or maze running) algorithm [5] is a widely used method for global and detailed routing. Briefly, the maze routing algorithm starts from a source point and recursively searches its neighbors for the best route until it reaches the sink point. The best route is defined by a function of congestion, wire length, chip size, number of bends, etc. Maze routing finds the optimal path for two-terminal nets according to the cost function. A major drawback of the algorithm is the large amount of memory required to label its data structure, the grid graph. There have been several other proposed extensions and modifications to the maze routing algorithm in the almost 40 years since it has been introduced, but the underlying method remains the same.

Pattern routing is the well-known idea of using prespecified patterns to route two terminal nets. This is particularly useful for high level computer-aided design (CAD) tools (i.e., tools preceding global routing in the design flow). For example, most placement tools use quick routing metrics to get a basic idea about congestion and wirelength information. In this paper, we develop quick routing methods that reduce the interconnect delay, increase the predictability of the circuit, and do not affect the quality of the routing solution. Since we know these metrics will not affect the routing, the placement tool can use these methods to model congestion and wirelength more accurately. Also, since we know the routing topology of a net, we can start wire sizing, wire planning, and optimally add buffers[1] once we have placement information.

Due to DSM effects, coupling is of greater importance for power, area, and timing in circuits. There are four principal reasons for this, increasing interconnect densities, faster clock rates, more aggressive use of high performance circuit families, and scaling threshold voltages [7]. In fact, coupling capacitance between wires can account for over 70% of the total wiring capacitance, even in 0.25-μm processes [8]. Therefore, it has become necessary to consider coupling during both global and detailed routing.

Until recently, there has been little research on the coupling problem in routing. Coupling reduction was considered at the detailed routing stage for the river routing problem [9], the channel routing problem [10], and the switch box routing problem [11]. Also, there have been efforts in reducing coupling in the stage between global and detailed routing. Xue *et al.* developed a post global routing tool which estimates the possible coupling between sensitive wires and tries to reroute nets away from possible crosstalk areas [12]. Chaudhary *et al.* developed a general postrouting spacing algorithm [13]. Also, coupling is examined for area routing [14] and global routing

Manuscript received July 12, 2001; revised January 2, 2002. This work was supported in part by the National Science Foundation DA program, UC-MICRO, Xilinx, and Fujitsu. This paper was recommended by Associate Editor M. D. F. Wong.

The authors are with the Computer Science Department, University of California, Los Angeles, CA 90095 USA (e-mail: kastner@cs.ucla.edu; elib@cs.ucla.edu; majid@cs.ucla.edu).

Publisher Item Identifier S 0278-0070(02)05628-2.

[1]If we know the net topology the complexity of buffer insertion for delay becomes polynomial time solvable [6].

[15]. This work presents algorithms for coupling avoidance routing. The algorithms are general so they can be used in both global and detailed routing.

In this paper, we focus on increasing routing predictability and reducing the unwanted effects caused by coupling during routing. This work is based on papers that appeared in [16]–[18]. In Section II, we give some basic definitions in order to make this paper self-contained. In particular, we discuss pattern routing, congestion, and coupling. Also, we briefly describe our router used in the experiments. Section III introduces the idea of increasing routing predictability through pattern routing. We present heuristics for finding a subset of nets which can be predictably routed and show the results of those heuristics. We introduce the coupling-free routing (CFR) problem in Section IV and discuss its applications to global and detailed routing. An exact algorithm for coupling-free routing decision problem (CFRDP) is presented. Then, we show how to transform a CFR problem into implication graph to model the dependencies between nets. Finally, we introduce the maximum coupling-free layout (MAX-CFL) problem and analyze a couple algorithms developed to solve the problem. We conclude in Section V.

II. Preliminaries

A *multiterminal net* $n = \{(x_1, y_1,), (x_2, y_2), \ldots, (x_n, y_n)\}$ is a collection of points in the plane. A *terminal* is single point of a net. A multiterminal net can be partitioned into a collection of *two-terminal* nets (a net with exactly two points) using a number of standard techniques. We adopt the *stable spanning tree* partitioning of Ho *et al.* [19]. Additionally, the spanning tree is altered for flexibility [20]. Essentially, we use this to transform the multiterminal net into a set of either a very short two-terminal net or a large two-terminal net. That paper shows that these nets can be pattern routed independently as two-terminal nets without affecting the routability of the circuit.

A two-terminal net (or simply called a *net* hereafter) $n = \{(x_1, y_1), (x_2, y_2)\}$ is an unordered pair of points (x_1, y_1) and (x_2, y_2). A *routing* or *wiring* of n is a set of horizontal and vertical line segments connecting (x_1, y_1) and (x_2, y_2). A *layout* is the routings of a set of nets.

A net n can be routed without any bends if and only if either $x_1 = x_2$ or $y_1 = y_2$. We call such a net a *zero-bend net*. Otherwise, there are two ways to route n with one bend as shown in Fig. 1. When a routing has no more than one bend, it is called a *single-bend routing* [21]. We call such a net a *one-bend net*.

The routings in Fig. 1 are called the *upper-L routing* and the *lower-L routing*. A stable spanning tree ensures that upper-L routing and lower-L routing shapes of the two-terminal nets obtained from a multiterminal net are pairwise nonintersecting. To avoid confusion, we often refer to a possible routing as a *route*. Thus we say that a one-bend net has two one-bend routes (the upper-L route and the lower-L route).

A *grid graph* is a graph $G(V, E)$ such that each vertex corresponds to a point in a plane. See Fig. 2 for further explanation. A *routing* of a net on a grid graph is a set of grid edges such that the terminals are fully connected. The *route edges* of a net are the set of edges used in the routing of that net.

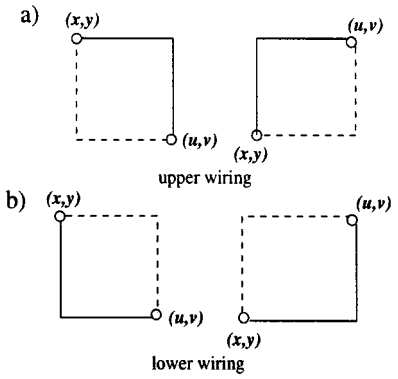

Fig. 1. (a) Upper-L routings. (b) Lower-L routings.

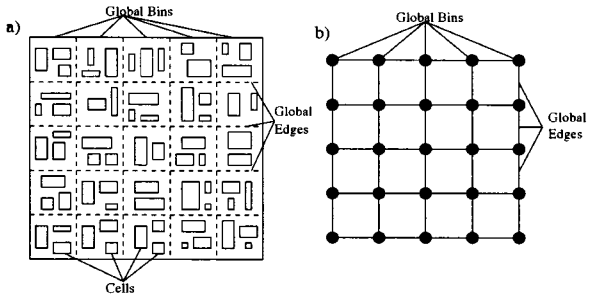

Fig. 2. (a) Placement of cells into global bins. (b) The corresponding grid graph.

A *global bin* is a rectangular partition of the chip. By partitioning the chip into many rectangular regions and placing the cells into these regions, we have a placement using global bins. The boundaries of the global bins are *global bin edges*.

In this paper, we assume that a global placement of cells and their interconnections are given by some placement engine (our experiments used Dragon [22]). The cells are placed into global bins and each cell is assumed to be placed in the center of the global bin. Looking at Fig. 2, it is easy to see that the global bins and edges can be transformed into a grid graph. The interconnections between the cells can be modeled by nets.

Congestion in a layout means that there are too many nets routed in a local area. This causes difficulty for the detailed router as it may not find a feasible routing solution. We want to evenly distribute the routing across the total chip area. The *congestion* of an edge is the number of nets routed over a global bin edge. From now on, we will refer to a global bin edge g as e_g. The *capacity* (also referred to as supply) of edge e_g is c_g. It is the maximum number of nets that can be routed over e_g. c_g is a fixed value that is based on the length of the edge and the technology used in creating the chip. The routing demand of e_g, specified as d_g, is defined as the number of route edges crossing e_g. Similarly, the demand of a vertex v is d_v. Here the demand corresponds to the number of routes that pass though the vertex v (equivalently the global bin v). An edge is overflown if and only if $d_e > c_e$. Formally, the overflow of an edge is

$$\text{overflow}_e = \begin{cases} d_e - c_e - t, & \text{if } d_e \geq c_e \\ 0, & \text{otherwise.} \end{cases}$$

t is a threshold value which allows d_e to go above c_e without an overflow penalty. t is used since you can often route up to t

nets though neighboring bins without affecting the congestion of those bins. t is usually a small constant (approximately 2–5). Using the global bin and global edge notation, the total overflow of a routing is

$$\text{overflow}_{\text{total}} = \sum_{e=1}^{|BE|} \text{overflow}_e$$

where BE is the set of bin edges. The total overflow reflects the shortage of routing resources for a particular set of edge capacities. A routing with a minimized total overflow is one of the objectives of our global router. Our industrial experience shows that total overflow is a good measure of congestion.

A. Maze Routing

We implemented a global maze router. The maze router takes every net and routes them one at a time according to a cost function

$$\begin{aligned}\text{overflow}_{\text{route}} &= \sum_{e \in \text{RouteEdges}} \text{overflow}_e \\ \text{length}_{\text{route}} &= |\text{RouteEdges}| \\ \text{cost}_{\text{route}} &= \alpha \times \text{overflow}_{\text{route}} + \text{length}_{\text{route}} \\ \text{cost}_{\text{total}} &= \sum_{\text{allnets}} \text{cost}_{\text{route}}.\end{aligned}$$

There is a tradeoff between minimizing overflow and minimizing wire length. Ideally, you could minimize both concurrently. Most often this is not possible. Our cost function can solely minimize wire length (set $\alpha = 0$). Likewise, you can minimize overflow by setting $\alpha \gg 1$. We found that varying α from 10 to 100 minimizes the total overflow while keeping the wire length minimal.

For nets with more than two terminals, we use stable Steiner trees to partition the net into a set of two-terminal nets. Each net is given an initial route and then a rip-up and reroute phase is applied to further minimize the total overflow. This technique (or variants of it) appears in most global routers in order to deal with the net ordering problem [23]. During rip-up and reroute, the bin edges are sequentially searched. If an edge is overflown, then all of the nets that pass through that edge are ripped and rerouted. This process continues until the total overflow converges to a local minimum. That is, if the total overflow does not decrease (the goal is to minimize the total overflow) after λ iterations, rip-up and reroute has completed. We found that a λ of 200 gave good results for the designs that we tested. Larger designs may need an increased λ which decreases the chance of getting stuck in a local minimum. In general, smaller designs can afford to decrease λ which would decrease the runtime.

B. Pattern Routing

Pattern routing is the notion of using predefined patterns to route two-terminal nets. Usually these are simple patterns such as an L-shaped (single bend) or a Z-shaped pattern with two bends, route restricted within bounding box. For more details, see Fig. 3.

Patterns can speed up the routing process. Instead of maze routing a net, we pattern route it. In general, maze routing will consider many bins that the final route will not actually use.

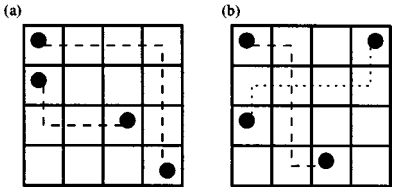

Fig. 3. (a) L-shaped routing of two two-terminal nets. (b) Z-shaped routing of two nets.

When using pattern routing, only a constant number of edges are searched. For example, L-shaped pattern routing will only search the edges on the bounding box of the two-terminal net. Then, depending on the cost of these edges, it will choose the upper-L or lower-L and place the route there. Similarly, Z-shaped pattern routing needs only search the edges on the perimeter and inside the two-terminal bounding box. On the other hand, maze routing will search every edge (in the worst case). Therefore, pattern routing has a better upper bound on runtime complexity. We found that on average, the pattern routing approach searches fewer edges than the maze router. We formally summarize the complexities.

1) Given a net $n = \{(x_1, y_1), (x_2, y_2)\}$ and a grid graph $G(V, E)$.
2) Let A be the edges on and within the bounding box of n. $A \subseteq E$. $|A| = 2{\cdot}|x_1{-}x_2|{\cdot}|y_1{-}y_2|{+}|x_1{-}x_2|{+}|y_1{-}y_2|$
3) Let P be the edges on the bounding box of n. $P \subseteq A$. $|P| = 2 \cdot (|x_1 - x_2| + |y_1 - y_2|)$
4) Maze routing—$O(|E|)$.
5) L-shaped pattern routing—$O(|P|)$.
6) Z-shaped pattern routing—$O(|A|)$.

Theorem 1: $|P| \leq |A| \leq |E|$.

Proof 1: The proof is trivial since $P \subseteq A \subseteq E$. □

The maze router ensures that the least cost route (according to the cost function) is found. Pattern routing does not give you this luxury. In fact, an L-shaped pattern routing could produce the second worst possible route. This occurs if both the upper-L route and the lower-L route are the two worst paths. Pattern routing will choose the better of these two solutions, giving you a bad routing. In general this is not the case, as our results show.

Another benefit of pattern routing lies in the predictability of a pattern-routed net [17]. If you know that a net will be pattern routed, you can quickly and accurately estimate its route earlier in the design flow. For example, you know that an L-shaped pattern route will take one of two routes. This allows higher level CAD tools, such as the placement or logic synthesis engines, to estimate routings which will lead to better congestion and area estimates. In order to exploit predictability, the tools need placement information. Many industrial logic synthesis tools are moving toward layout-driven synthesis. Additionally, an academic behavioral level synthesis tool has recently incorporated placement information [24].

With emergence of deep submicron (DSM) fabrication technology, interconnect has an increasingly dominant role. Now circuit delay is determined by the gate resistance and capacitance as well as the interconnect resistance and capacitance [25]. When optimizing for delay in a circuit, logic synthesis tools look at the critical path. Usually these tools only consider the gate delay, ignoring the interconnect delay. If we could pattern route

the gates on the critical path, then we can more accurately estimate the interconnect resistance and capacitance.

Finally, the number of vias on a pattern-routed net is fixed. Since vias further increase the capacitance and resistance, it is beneficial to keep them at a minimum. Also, vias negatively affect the routability of the circuit [26].

C. Coupling

Bakoglu [27] shows that the wire-delay on a distributed RC line contains a $R_W \cdot (C_S + C_C)$ time constant, where R_W is the interconnect resistance and C_S and C_C are the substrate and coupling capacitances

$$R_W \cdot (C_S + C_C) = \frac{\rho \cdot l}{w \cdot t}\left(\frac{\varepsilon_\kappa \cdot l \cdot w}{h} + \frac{\varepsilon_\kappa \cdot l \cdot t}{s}\right) \quad (1)$$

where ρ is resistivity of the conductor, ε_κ is the insulator dielectric constant, and w, t, and h are the conductor's width, thickness, and separation from the substrate, respectively. The terms l and s are the coupled length and spacing of the interconnect.

The coupling capacitance C_C between two wires i and j can also be represented as follows:

$$C_C(i,j) = \frac{f_{ij} \cdot l_{ij}}{\text{dist}_{ij}} \frac{1}{1 - \frac{w_i + w_j}{2d_{ij}}} \quad (2)$$

where w_i and w_j are the sizes of wires i and j $(w_i, w_j > 0)$, f_{ij} is the unit length fringing capacitance between wires i and j, l_{ij} is the overlap length of wires i and j and dist_{ij} is the distance from the center line of wire i to the center of wire j (see Fig. 4).

We are trying to minimize the coupling. During routing, we can control l_{ij}, dist_{ij}, w_i, and w_j. By avoiding overlap between two wires, l_{ij} can be minimized. In other words, we do not want adjacent wires to run in parallel for long distances. We assume that w_i, w_j, l_{ij} are fixed; we do not consider wire sizing and spacing in our algorithm. But, this can be done as a postprocessing step using a number of techniques (see [28] ad [29] for a comprehensive survey and tutorial).

There are two problems introduced by coupling, delay deterioration, and crosstalk. Delay deterioration refers to the fact that the total capacitance seen by a gate is no longer a constant value. The rising contribution of coupling capacitance to total load capacitance makes the Miller effect evident. Delay deterioration occurs because the Miller effect causes the capacitance to vary. For example, if two coupled nets switch in opposite directions at the same time, the capacitance, hence the delay, will increase.

Crosstalk is a type of noise [2] introduced by coupling between two adjacent wires. A change in voltage or current on one of the wires may interfere with the signal on the other wire. There are two unwanted effects of crosstalk. First, the two wires form a mutual inductor. This inductive effect must be considered as circuit frequencies move above 500 MHz [30], [31]. Inductive effects are not addressed in this work. The second effect is associated with coupling capacitance. Coupling capacitance can cause a switching net to induce noise onto a neighboring net possibly resulting in an incorrect functional response.

Coupling between nets is not always detrimental. In [32], Kirkpatrick and Sangiovanni-Vincentelli introduce the notion of *crosstalk constraint generation* which uses the concepts of *analog and digital sensitivity* and a physical coupling term in order to reduce the constraints given to layout synthesis. This allows us to remove false crosstalk constraints. For example, a net A may couple with net B. But, net A could have a high tolerance for delay and noise. Therefore, the A and B can couple without negative circuit performance. We want to remove these cases as they unnecessarily over-constrain the problem.

[2]Noise is defined as an unwanted variation which makes the behavior of a manufactured circuit deviate from the expected response.

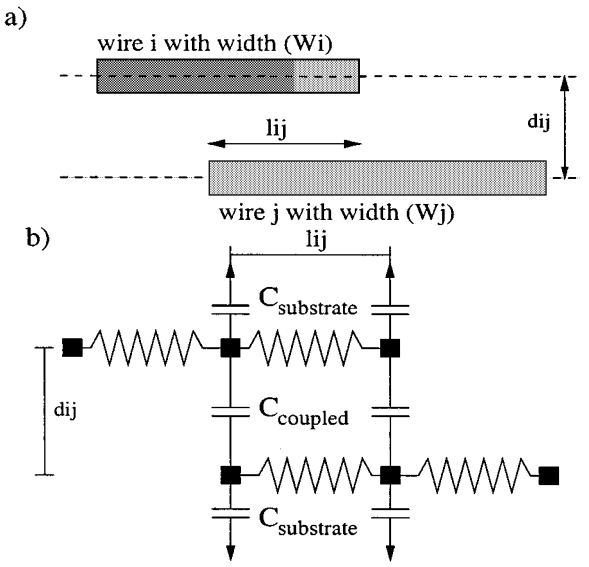

Fig. 4. (a) Physical coupling capacitance between two wires. (b) The wires modeled by resistors and capacitors.

III. Using Patterns While Maintaining Routability

In this section, we show the effect of pattern routing on the quality of the routing solution. We show that you can pattern route up to 80% of the nets with smallest bounding boxes while incurring little or no loss of quality. Then, we show how a set of nets that satisfies the γ-density routing problem (formally defined in Section III-C) can be pattern routed without sacrificing the routing quality. This gives us the ability to pattern route a subset of all the nets, even if the nets have a large bounding box.

A. Benchmarks

To perform our experiments, we used five MCNC standard-cell benchmark circuits [33]. The characteristics of the circuits are shown in Table I. The circuits were placed into global bins using the Dragon global and detailed placement engine [22]. Some of the benchmarks (i.e., prim1 and prim1.2) are repeated. Repeated benchmarks differ in the number of global bins; they consist of the same number of nets, cells, and pins but may have a completely different placement.

B. Pattern Routing Analysis

For our experimental results, we choose to use L-shaped pattern routing over Z-shaped for a several reasons. First, for two-terminal nets there are only two possible L-shaped routes to consider. The number of Z-shaped routes grows linearly with the bounding box size. Since we are aiming toward predictable routes, L-shaped patterns reduce the choices of routings. Secondly, we want the routing to execute quickly. The time to find

TABLE I
BENCHMARK CIRCUIT INFORMATION

Data file	Num Cells	Num Nets	Num Pins	Global Bins
prim1	833	1156	3303	8 X 16
prim1.2	833	1156	3303	16 X 16
prim2	3014	3671	12014	8 X 16
prim2.2	3014	3671	12014	32 X 32
avqs	21584	30038	84081	30 X 80
avqs.2	21584	30038	84081	80 X 80
biomed	6417	7052	22253	20 X 40
biomed.2	6417	7052	22253	40 X 40
struct	1888	1920	5407	20 X 16

the congestion of the routes is $O(|P|)$, whereas the Z-shaped routes is $O(|A|)$. Theorem 3.1 states the $|P| \leq |A|$.

We comment on a few observations. Even though pure maze routing has the greatest freedom in terms of finding the least congested solution, the overall algorithm is a heuristic, therefore it is not guaranteed to find the optimal solution. The tradeoff between fast routing time and reduced number of routings (better predictability) favors L-shaped routing. Therefore, we will exclusively use L-shaped routing for all of our pattern routing experiments.

Our experiments focused on determining which nets we can pattern route while incurring little to no congestion penalty. Our first heuristic (referred to as the largest first pattern route (LFPR) heuristic) splits the multiterminal nets into two terminal nets and sorts them from largest bounding box to smallest bounding box. Then, we pattern routed the $x\%$ largest nets while maze routing the rest of the nets. The pattern routed nets were not rerouted during the rip and reroute phase. As shown in Table II, pattern routing large nets gives unfavorable overflow results. If you pattern route only the largest 5% of the nets, your overflow increases more than two-fold over maze routing every net. A similar trend occurs as you increase the pattern route percentage. Pattern routing only 20% of the nets results in an overflow over four times the 0% overflow. (Note, the 0% pattern route is exactly equivalent to maze routing every net; the rip and reroute stage will consider every net.)

The smallest first pattern route (SFPR) heuristic gives more encouraging results. This heuristic is similar to LFPR except here we sort the two terminal nets from smallest to largest. Thus, an SFPR of 5% will pattern route the smallest 5% of the nets. Referring to Table III, we can see that we can pattern route up to 80% of the nets with only a small increase in overflow. In fact, pattern routing the small nets actually leads to better overflow results! These results add validity to our previous statement that pattern routing can lead the maze router to better overflow solution.

This SFPR heuristic results may seem surprising. Looking at Table IV, you can see the percentage of the total route length that the smallest $x\%$ of the nets comprises. Even when you pattern route the smallest 90% of the nets, the route length of these small nets is, on average, only 58.32% of the total route length. This means that the remaining 10% of the nets that are maze routed are much longer than the short nets. This allows the maze router enough freedom to find a good routing, even when

TABLE II
CONGESTION DATA FOR LARGEST FIRST-PATTERN ROUTE HEURISTIC. 0% IS THE BASE CASE CONGESTION. THE REMAINING RESULTS TAKE THE CONGESTION AND SUBTRACT THE BASE CASE CONGESTION. SO 40 (AS IN PRIM1 AT 5%) MEANS A TOTAL CONGESTION OF 165+ 40= 205

Datafile	0%	5%	10%	15%	20%
prim1	165	40	64	82	91
prim1.2	121	46	79	88	99
prim2	112	26	71	101	190
prim2.2	35	-3	44	69	142
avqs	63	224	384	414	464
avqs.2	18	394	575	670	730
biomed	25	43	269	366	386
biomed.2	47	-5	238	363	408
struct	74	120	182	228	252
total	**660**	**885**	**1906**	**2381**	**2762**

TABLE III
CONGESTION DATA FOR SMALLEST FIRST-PATTERN ROUTE HEURISTIC. 0% IS THE BASE CASE CONGESTION. THE REMAINING RESULTS TAKE THE CONGESTION AND SUBTRACT THE BASE CASE CONGESTION. A NEGATIVE RESULT MEANS THAT THE CURRENT CONGESTION IS BETTER THAN THE BASE CONGESTION

Datafile	0%	50%	60%	70%	80%	90%
prim1	165	-4	0	-2	-3	6
prim1.2	121	0	-3	11	9	5
prim2	112	1	1	-1	4	28
prim2.2	35	-1	-4	-5	-4	-14
avqs	63	-14	-5	6	15	27
avqs.2	18	7	-3	12	0	6
biomed	25	0	-2	-1	0	5
biomed.2	47	0	-3	-2	4	9
struct	74	-3	6	13	32	52
total	**660**	**-14**	**-13**	**31**	**57**	**124**

90% of the nets are fixed. This gives some insight as to why the LFPR heuristic does not work. If you fix the long nets to a pattern, you greatly reduce the routing freedom that the maze router needs to produce a good route. Since the small nets are close in physical proximity, there are limited number of routes that these nets could take. Therefore, the maze router may find a less congested solution, but due to the small number of feasible routes, the pattern route solution will not significantly vary from the best (i.e., maze-routed) solution. Additionally, small nets are often entirely located within a congested region. In this case, any shortest length path will be essentially equivalent in terms of overflow minimization. Since there is no quality improvement using maze routing, the pattern route is preferable due to its faster run time and predictability.

We have shown that you can pattern route up to 80% of the nets with small bounding boxes. Unfortunately, you cannot do pattern routing on nets with large bounding boxes using the LFPR heuristic without suffering a huge loss in the quality of solution. Now, we will show that any set of γ-density nets can be pattern routed without degrading the solution quality. This allows us to pattern route the nets with large bounding boxes.

C. γ-Density Routing

The *γ-density (γ-d) routing* problem tries to find a one-bend routing of two-terminal nets such that the routing demand of

TABLE IV
PERCENTAGE OF ROUTE LENGTH USED BY SFPR NETS. FOR EXAMPLE, WHEN YOU PATTERN ROUTE THE 10% SMALLEST NETS IN PRIM1, THE ROUTE LENGTH OF THOSE NETS IS ONLY 5.75% OF THE TOTAL ROUTE LENGTH

Datafile	10%	20%	30%	50%	80%	90%
prim1	5.8	11.5	17.3	28.5	50.7	64.1
prim1.2	5.6	11.2	16.9	28.1	54.7	69.4
prim2	6.3	12.6	19.0	31.6	52.2	65.2
prim2.2	3.7	7.4	11.1	18.5	41.0	54.7
avqs	2.8	5.5	8.3	13.9	32.8	49.1
avqs.2	3.6	7.2	10.9	18.1	36.6	50.5
biomed	3.6	7.3	10.9	18.1	40.8	54.8
biomed.2	2.9	6.0	9.0	14.9	36.3	45.0
struct	3.3	6.6	9.9	21.9	52.3	67.1
avg	**4.2**	**8.4**	**12.6**	**21.5**	**44.1**	**58.3**

every bin edge is less than γ. Let us define the γ-density routing problem formally.

1) Given a set of two-terminal nets N, a grid graph $G(V, E)$, and an integer γ.
2) Does there exist a one-bend routing for every net $i \in N$ such that $d_e \leq \gamma$ for every edge $e \in E$?

In Table V, we show that pattern routing on a set of one-dimensional (1-D) or 2-D routable nets does not affect the overall routing solution quality. Since we are trying to show that nets with large bounding boxes can be pattern routed, we used a heuristic that focused on finding such nets. Like the LFPR heuristic, we sort the nets from largest to smallest bounding box. Then, we assign an upper or lower routing to the nets so that they can be γ-d routed. Therefore, some of the largest nets are always in the set of γ-d nets. Table V also shows the overflow results when we pattern route a set of 1-D, 2-D, 3-D, 4-D, and 5-D nets. Notice that some circuits allow up to 5-D routing without loss of quality. This highly depends on the number of nets and number of bins in the benchmark. For example, avqs is a large benchmark and the nets in the 3-D routing only account for 17.7% of the total routing. Compare this to prim1.2 where the nets of the 3-d routing are 35.5% of the total routing. Notice that 1-D routing does not hurt the solution quality for all but three benchmark (here avqs seems to be an anomaly since the 2-D, 3-D, and 4-D routings show no degradation of the overall routing quality). We believe that as the capacity of the edges grows larger, the allowable density (value of γ) can increase while maintaining similar routability.

IV. USING PATTERNS TO REDUCE COUPLING

In Section III, we showed that it is possible to use L-shaped patterns to routes some nets without affecting the quality of the routing solution. By pattern routing the nets, we reduce their interconnect delay (since the wirelength is minimal). But, coupling is also an important component of delay that must be considered. Therefore, we need methods to reduce the coupling between nets. In this section, we present some theoretical aspects to reduce coupling between nets and introduce some algorithms to implement this theory. The ideas that we introduce provide a framework from which more complex algorithms and methods can be derived. We discuss some possible derivations to both the global and detailed routing problems.

TABLE V
OVERFLOW INFORMATION FOR PATTERN ROUTING A SET OF γ-d NETS. γ IS VARIED FROM 1—5. THE BASE CASE IS THE TOTAL OVERFLOW WITH PURE MAZE ROUTING. THE NEXT COLUMNS ARE CURRENT OVERFLOW—BASE CASE. A LOWER VALUE MEANS BETTER OVERFLOW, HENCE A BETTER SOLUTION

Datafile	base	1-d	2-d	3-d	4-d	5-d
prim1	165	-2	5	-5	-7	-7
prim1.2	121	0	6	6	3	3
prim2	112	0	-4	4	-3	2
prim2.2	35	8	-2	0	-6	2
avqs	63	10	-16	-3	-7	-6
avqs.2	18	-13	-2	6	-4	22
biomed	25	-2	-1	4	-5	1
biomed.2	47	-6	2	-4	11	6
struct	74	6	10	10	10	10
total	**660**	**1**	**-2**	**18**	**-8**	**33**

A. Coupling-Free Routing

Every route consists of horizontal and/or vertical line segments. We say two wires *couple* if the line segments forming them are closer than d units for more than l units. Two line segments *intersect* if they have at least one point in common and *overlap* if they have more than one point in common.

For a given set of nets $S = \{n_i = \{(x_{1i}, y_{1i}), (x_{2i}, y_{2i})\} \mid 1 \leq i \leq |S|\}$, a (single-bend) layout of S is coupling-free if there are no two routes that run in parallel at a distance equal to or closer than s units for more than l continuous units. Examples of coupled and noncoupled layouts are given in Fig. 6. Given a set of two-terminal nets, the problem of obtaining a coupling-free routing of nets is called the *coupling-free routing problem* (CFR problem). A more complex formulation to decide coupling can be substituted in lieu of our coupling-free definition. For example, we can use a complex coupling equation, e.g., (2), and define two nets as coupling-free if they have a coupling capacitance less than some threshold value. The theory and equations we present will hold for any pairwise definition of coupling.[3] Additionally, it is straightforward to extend the formulation to consider the cumulative coupling effect caused by multiple neighboring nets. Consider the situation in Fig. 5. When considered separately, the lower-L routing of two nets A and B do not couple with upper-L routing of Net C. But, when both A and B are routed in a lower-L the additive effect of the coupling causes a coupling violation for the upper-L routing of C. We will explain how to handle such cases. Unfortunately, by considering these cases, the complexity of the problem substantially increases.[4]

We consider routing only a subset of nets for a few reasons. First, by routing a subset of the critical nets as patterns, we guarantee that the nets have the minimum wirelength, which reduces the interconnect delay of the nets so that the timing constraints can be met. The remaining critical nets can be routed using other more general coupling aware routing techniques, e.g., maze routing that considers coupling and timing as in [34]. We are presenting a fundamental algorithm with polynomial

[3]Note: the runtime may increase due to increased complexity of coupling calculation.

[4]We go from solving 2-SAT to solving the general SAT problem, which is NP-complete.

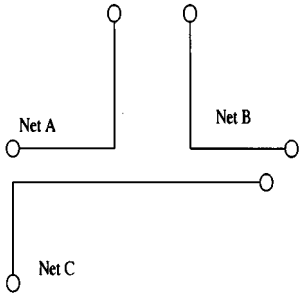

Fig. 5. The combination of two routings cause a noncoupling-free layout.

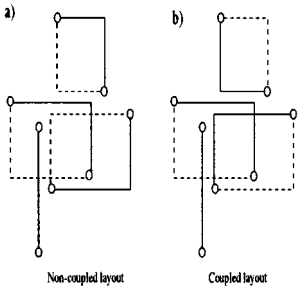

Fig. 6. (a) Coupling-free routings. (b) Noncoupling-free routings.

runtime and basic theoretical properties. Additional heuristics can easily be added onto this algorithm to increase its application. We believe that a solid framework with fundamental properties is needed for every heuristic [35]; this paper presents a basic coupling algorithm to which heuristic extensions can be added. Now, we discuss some possible applications and extensions that may be added to our base algorithm.

As very large scale integration (VLSI) fabrication technology progresses, more routing layers become available. Therefore, we can afford to set aside *preferred layers* for critical nets. A preferred layer usually has a lower wiring resistance due to position of the layer (lower layers have lower resistance) and width of the wires on that layer (large wire widths have lower resistance). Power, ground, and clock nets are already routed on preferred layers. We propose using the preferred layers for routing critical nets. Critical nets are allotted very little slack in order to meet timing constraints. Since interconnect is becoming a dominate factor in delay of a circuit and coupling plays are large role in interconnect delay, these nets should be routed in order to minimize coupling and wirelength. Therefore, we can use notion of coupling-free routing to provide a detailed routing for the critical nets. Since the nets are routed with at most one bend, they have minimum wirelength. In addition, coupling-free routing minimizes the coupling of the routed nets. Combining these two factors, we have a routing of the critical nets with minimal interconnect delay. After we have a coupling-free layout, noncritical nets can be routed, using any type of routing method, e.g., maze routing, on the preferred layers to maximize routing resources. Additionally, we can consider all minimum length routes, e.g., z-shapes. It is possible to extend our algorithms to consider z-shapes, though this extension creates a dramatic increase in complexity.[5]

[5]Once again, the formulation goes from 2-SAT to the general SAT problem.

Many single-layer routing algorithms have been suggested. Liao *et al.* [36] propose density routing or maze routing to perform this task. A more recent paper by Lin and Ro [37] improves on the work by Liao *et al.* They employ a two step process. First, they find a planar set of single-bend nets without considering coupling. Then, they use a method based on rubberband equivalent to find a routing for the remaining nets. CFR can easily be incorporated into the first stage of Lin and Ro's algorithm to obtain a planar layout that is coupling-free.

Generally, coupling at the global routing stage is hard to determine. A global route is not exact. Therefore, a net could possibly couple with every net that is routed in the same global bin. But, the net will only couple with its immediate neighbors.[6] Ultimately, track assignment (which can be done at the global or detailed routing stage) determines the coupling. Additionally, the detailed router will often make local changes which can affect the coupling of nets [38]. But, the detailed router can only make local changes, therefore considering coupling at the global stage, even if it is not exact, is beneficial as it can provide a way to make large scale changes to a layout that otherwise cannot be done at the detailed level. If we have coupling-free layout at the global stage, then the layout will remain coupling-free at the detailed stage. Therefore, we can use CFR at the global routing stage to reduce coupling for the detailed router. This is similar to wire planning; we are trying to find a general area for the net's routing. Then, the detailed router can consider more exact coupling while making track changes, locally permuting the wiring (adding additional bends) and changing the spacing between wires as in [39]. Additionally, we could "freeze" the routings at the detailed level to insure that they remain coupling-free.

Next, we propose an exact algorithm for determining if a set of nets can be a coupling-free routing. Then, we describe a couple heuristics for solving the *maximum coupling-free layout problem* — the maximum number of nets that can be laid out in a coupling-free fashion.

B. The Coupling-Free Routing Decision Problem

Given a set of two-terminal nets S, is there a single-bend routing for every net in S such that no two routings couple? That is, do there exist any routes that run in parallel at a distance equal to or closer than s units for more than l continuous units?

We solve the coupling-free routing decision problem by transforming it into an instance of the 2-satisfiability (2-SAT) problem.

The 2-Satisfiability Problem: Given a set U of variables, a collection C of clauses such that each clause $c \in C$ has $|c| = 2$. Is there a satisfying truth assignment for U?

The 2-SAT problem can be solved in $O(|U|)$ time [40].

In order to transform an instance of CFR decision problem to 2-SAT, we assign a boolean variable to each net. Without loss of generality, we say if net A has an upper-L route if its variable is true (x_A) and a lower-L route if its variable is false $(\overline{x_A})$. A routing of a net may *force* a routing of another net. For example, assume net A is routed in an upper-L. If the upper-L routing

[6]Theoretically, a net couples with every net on the chip. But, the neighboring nets act as a shield which makes the coupling capacitance seen by the other nets minimal.

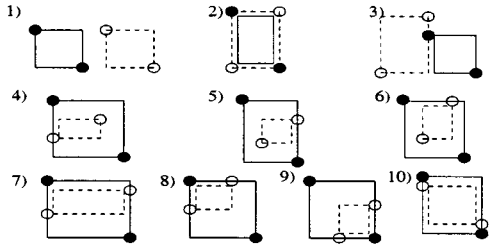

Fig. 7. Examples of the ten interactions for the coupling-free routing problem. The solid points and lines correspond to net A. The dotted lines and circles correspond to the the bounding box and terminals of net B, respectively.

of A (x_A) couples with the lower-L routing of B ($\overline{x_B}$), then net B must be routed as an upper-L to avoid coupling. Hence x_A forces x_B. With respect to two nets A and B, there are ten possible forcing interactions between these nets.

1) A and B are independent. Either layout for each net does not directly influence the layout for the other.
2) A and B cannot be couple-free routed.
3) The lower-L routing for A forces the upper-L routing for B. However, the upper-L routing for A does not influence the routing of B. The next three cases are similar.
4) The lower-L routing of A forces the lower-L routing of B.
5) The upper-L routing of A forces the upper-L routing of B.
6) The upper-L routing of A forces the lower-L routing of B.
7) The lower-L routing of A forces a lower-L routing of B. Also, the upper-L routing of A forces an upper-L routing of B. The next three cases are similar to this case.
8) Lower-L of A forces lower-L of B; Upper-L of A forces lower-L of B.
9) Lower-L of A forces upper-L of B; Upper-L of A forces upper-L of B.
10) Lower-L of A forces upper-L of B; Upper-L of A forces lower-L of B.

Examples of all of these cases are given in Fig. 7.

The algorithm proceeds as follows.

Stage 1: Consider the $(|S|(|S|-1)/2)$ interactions where S is the set of nets under consideration. If two nets cannot be couple-free routed (corresponding to interaction 2), the algorithm terminates and returns FALSE. For each pair of nets, i and j, we determine the interaction between n_i and n_j. Using this information, we can determine which wires are forced.

Stage 2: The constraint information must be encoded into boolean expression with these properties:

1) It is in conjunctive normal form (CNF) (see [41])
2) It contains at most two literals per clause
3) It is satisfiable if and only if the corresponding wire set can be laid out (without coupling) in a single bend fashion.

Each of the ten interactions can be encapsulated as a binary relation.

1) A and B are independent. No encoding
2) A and B can not be couple-free routed. No encoding, the algorithm will terminate and return FALSE if this case is found.
3) The lower-L routing for A forces the upper-L routing for B. Encoded as $(x_A \vee x_B)$
4) The lower-L routing of A forces the lower-L routing of B. Encoded as $(x_A \vee \overline{x_B})$
5) The upper-L routing of A forces the upper-L routing of B. Encoded as $(\overline{x_A} \vee x_B)$
6) The upper-L routing of A forces the lower-L routing of B. Encoded as $(\overline{x_A} \vee \overline{x_B})$
7) The lower-L routing of A forces a lower-L routing of B. Also, the upper-L routing of A forces an upper-L routing of B. Encoded as $(x_A \vee \overline{x_B}) \wedge (\overline{x_A} \vee x_B)$
8) Lower-L of A forces lower-L of B; Upper-L of A forces lower-L of B. Encoded as $(x_A \vee \overline{x_B}) \wedge (\overline{x_A} \vee \overline{x_B})$
9) Lower-L of A forces upper-L of B; Upper-L of A forces upper-L of B. Encoded as $(x_A \vee x_B) \wedge (\overline{x_A} \vee x_B)$
10) Lower-L of A forces upper-L of B; Upper-L of A forces lower-L of B. Encoded as $(x_A \vee x_B) \wedge (\overline{x_A} \vee \overline{x_B})$

For each forced wire A, if the wire is forced to an upper-L route, this is encoded as x_A; if the wire is forced to a lower-L route, this is encoded as $\overline{x_A}$.

Every net n is given a boolean variable. Therefore, $|U| = |S|$. The entire set of $(|S|(|S|-1)/2)$ interaction relations are encoded as specified. Each of these relations becomes a clause in the 2-SAT instance.

Lemma 1: $|C| = O(|S|^2)$.

Proof 1: Since there are at most two relations per interactions, $|C| \leq |S|\,(|S|-1)$. □

The 2-SAT instance is obtained by letting each net n be a boolean variable $\in U$. The set of clauses C are the encoded net interactions.

Theorem 2: The coupling-free routing decision problem can be solved in $O(|S|^2)$ time.

Proof 2: The CFRDP $\propto$ 2-SAT in $O(|S|^2)$ time. An instance of 2-SAT can be solved in linear time. Therefore, we can solve the coupling-free routing decision problem in $O(|S|^2)$ time. □

If we want to consider the cumulative effect of coupling between a set of nets, we can add additional clauses to the 2-SAT formulation we have just described. First, we must identify the set of nets that cumulatively cause a coupling violation as in Fig. 5. For each case, we add an additional clauses and variables. The clauses added will have a cardinality greater than 2, i.e., we will no longer have a 2-SAT formulation. For the example in Fig. 5, we add two additional clauses c_1 and c_2 and one additional variable x_D as follows: $c_1 = (x_A \vee x_B \vee x_D), c_2 = (\overline{x_D} \vee \overline{x_C})$. The new variable x_D indicates if both nets A and B are routed in a lower-L fashion. If that is the case, clause c_2 forces net C to be routed as a lower-L to avoid the joint coupling effect. The additional clauses and variables for other cases can be derived in a similar manner.

C. Implication Graph

In this section, we show how an instance of the CFR problem is transformable into an *implication graph*. Then, we define

some properties associated with the implication graph. We can utilize the properties of the implication graph to solve the CFR problem.

D. 2-SAT $\propto$ Implication Graph

First, we show how an instance of 2-SAT is transformable into an implication graph. In Section IV, we show how to transform an instance of the CFR problem to an instance of 2-SAT. Since CFR $\propto$ 2-SAT $\propto$ implication graph, CFR $\propto$ implication graph. The multistep transformation allows us to elegantly prove many properties associated with the implication graph. But, we will also show how to directly transform the CFR problem to an implication graph.

Let $C = C_i(x_i \vee y_i)$ be an instance of 2-SAT, where x_i, y_i are literals over $u_1, \ldots, u_n \in U$. We want to know when SAT(C) is true. Define a digraph $G = (V, E)$ by letting V be the set of literals and $(x, y) \in E$ if and only if $\overline{x} \vee y$ is one of the clauses. Recall that $\overline{x} \vee y$ is equivalent to $x \Rightarrow y$ (implication). We can assume there is no clause of the form $x \Rightarrow x$ since that is always true. Finally, note that $x \Rightarrow \cdots \Rightarrow y$ implies $x \Rightarrow y$.

Theorem 3: If there is a cycle in G containing both x and $\overline{x}$ for all $x \in V$, C is not SAT.

Proof 3: The reason is that if $x \Rightarrow \overline{x}$, then x must be false. But since there is a cycle $\overline{x} \Rightarrow x$ which means x must be true. We have a contradiction. Therefore, C is SAT iff G does not contain any cycles including x and $\overline{x}$ for any literal x. □

We call the digraph G an implication graph since it models the implications between the literals.

E. Coupling-Free Routing $\propto$ Implication Graph

Now we show how the CFR problem is directly transformable into an implication graph.

Given a set of nets N. The implication graph is a directed graph (digraph) $G_{\text{imp}}(V, E)$. Let every vertex $v \in V$ correspond an upper-L routing and lower-L routing of each net $n \in R$. Therefore, $|V| = 2 \times |N|$. Then, $(x, y) \in E$ if and only if x forces y or, equivalently, $x \Rightarrow y$. We call this an implication.

Theorem 4: If there is an implication $x_A \Rightarrow x_B$, there is contrapositive implication $\overline{x_B} \Rightarrow \overline{x_A}$.

Proof 4: Since $x_A \Rightarrow x_B$, the upper-L routing of x_A must couple the lower-L routing of x_B. Therefore, a lower-L routing of net B ($\overline{x_B}$) will force a lower-L routing of net A ($\overline{x_A}$). □

Theorem 5: Given a set of nets N, the construction of the corresponding implication graph takes running time $O(|N|^2)$.

Proof 5: First, we must determine the forcing interactions between every net. There are $(|N|(|N|-1)/2)$ possible interactions. Determining whether coupling exists in each interaction take $O(1)$ time. Therefore, it takes $O(|N|^2)$ time to determine the interactions. The number of vertices in the implication graph is exactly $2|N|$. The maximum number of edges is

$$\binom{2|N|}{2} = O(|N|^2).$$

The forcing interactions determine whether or not an edge exist. This requires a simple $O(1)$ lookup into an interaction table. Adding up the complexities gives us the runtime of $O(|N|^2)$. □

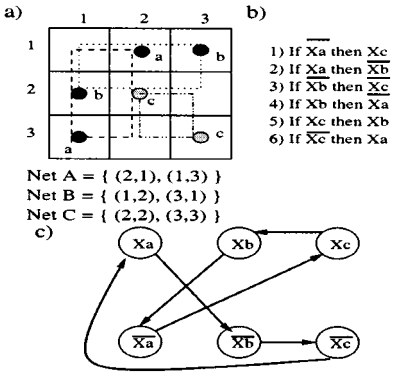

Fig. 8. (a) The layout of nets A, B, and C. (b) The implications of the nets. (c) The implication graph. x_i indicates an upper-L routing of net i. The implication graph does not have any cycles containing x_i and $\overline{x_i}$, $i \in A, B, C$; therefore, the nets are coupling-free routable.

Lemma 2: Consider a set of nets N and its corresponding implication graph G. If there is a cycle in G containing x_i and $\overline{x_i}$ where $i \in N$, then the nets N are not couple-free routable.

Proof 2: This is a direct consequence of Theorem 4.1. This should not be surprising since we can transform the CFR problem into 2-SAT. □

Lemma 3: Given a set of nets N, there is an $O(|V||E|)$ algorithm to determine if these nets are coupling-free routable.

Proof 3: Theorem 5 says that an implication graph is created in $O(|N|^2)$ time. According to Lemma 2, if we find a cycle containing x_i and $\overline{x_i}$ the nets N are not coupling-free routable. We can look for these cycles by doing a depth-first search from every vertex. If there is a path from x_i to $\overline{x_i}$ and a path from $\overline{x_i}$ to x_i, there is a cycle containing x_i and $\overline{x_i}$. We can do this for every vertex in $O(|V||E|)$. $O(|N|^2) \leq O(|V||E|)$. Therefore, we can determine if the nets are coupling-free routable in $O(|V||E|)$. □

For each implication case, up to two clauses are added to 2-SAT in the transformation. These clauses correspond directly to edges in the implication digraph. Fig. 8 shows a simple example for three nets. Focusing on nets A and B, we see that an upper-L routing of net A forces a lower-L routing for net B (corresponding to case 6). Therefore, we add the clause $(\overline{x_A} \vee \overline{x_B})$ to the 2-SAT instance. In the implication graph, we add an edge from vertex x_A to vertex $\overline{x_B}$. Notice that an upper-L routing of net B forces a lower-L routing of net A. This corresponds to $x_B \Rightarrow \overline{x_A}$ which is the contrapositive of the previous statement. The other cases are similar. Notice that there are no cycles in the implication graph in Fig. 8(c). This means that these three nets can be coupling-free routed.

F. Properties

1) Direct Forcing: Assume that we have implication graph $G_{\text{imp}}(V, E)$ which is constructed from an instance of a CFR problem containing the set of nets N. Remember that every vertex in the implication graph corresponds to a routing of a net $\in N$. Therefore, there are two vertices per net, one vertex for the upper-L routing and one vertex for the lower-L routing. We define the routing corresponding to vertex v as *route*(v). Let $u, v \in V$ be two unique vertices. If there is a directed edge

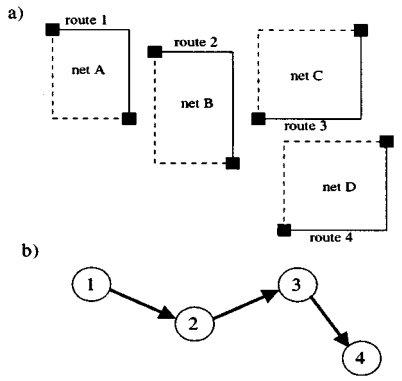

Fig. 9. (a) The layout of the nets. (b) The implication graph for routes 1, 2, 3, and 4.

(u, v), then the route (u) forces route (v). This is a direct consequence of the way that the implication graph is constructed.

The *outdegree* of vertex v in a digraph is the number of vertices adjacent to v. In an implication graph, the outdegree of v corresponds to the number of routings that route (v) forces. We call this a *direct forcing*.

2) Indirect Forcing: A routing may force a net even if it is not a direct forcing. Referring to Fig. 9, Route 1 directly forces only one route, Route 2. But, Route 2 forces Route 3 which forces Route 4. So, if we choose to route the net A in an upper-L manner (Route 1), then nets B, C, and D must be laid out as Routes 2, 3, and 4, respectively, if we want to route every net. Route 1 forces three routes even though it only directly forces Route 2. We say that Route 1 indirectly forces Routes 2, 3, and 4.

Given an implication graph $G_{\text{imp}}(V, E)$ and vertices $u, v \in V$. A v *indirectly forces* u if there is a path from u to v. The number of total forcings (direct and indirect) of v is calculated by determining the number of vertices that are connected to v. A slightly modified version of depth-first search can be used to determine the number of indirect forcing in time $O(|V|)$.

G. Maximum Coupling-Free Layout

The Maximum Coupling-Free Layout Problem (MAX-CFL): Given a set of two-terminal nets S and a positive integer $K \leq |S|$. Is there a single-bend routing for at least K nets in S such that no two routings couple?

Theorem 6: The maximum coupling-free layout problem for planar layouts is NP-Complete.

Proof 6: We make a transformation from the MAXWIRE problem. The MAXWIRE problem is defined as finding a subset of nets T where $T \subseteq S$ and $|T| \geq K$ such that all the wires in T can be laid out in a single bend fashion on one layer. The MAXWIRE problem is NP-Complete [42]. By setting the coupling variables $d = \infty$ and $l = 0$, we can directly transform any instance of MAXWIRE to an instance of MAX-CFL. This essentially removes any coupling restrictions from the problem. □

MAX-CFL can be extended for consideration criticality. The criticality of a net can be defined in numerous ways. Most often, a net's criticality is determined by the amount of timing slack that is available to that net. Also, the length of a net can be used. If we consider criticality, MAX-CFL tries to route a subset of nets with maximum criticality. A subset with maximum criticality will not always be the subset of maximum size.

Additional routing restrictions to the MAX-CFL problem are often needed. For example, we can use MAX-CFL to find a subset of planar nets. In this case, we must slightly modify the algorithms to consider intersection between the nets. Another common routing problem allows two layers to route the nets—one for vertical segments, one for horizontal segments. In this case, we must consider overlap between the nets. The algorithms that we present next assume that there are no restrictions. With the proper simple modifications, they can consider such restrictions.

Now, we look at a few heuristics to solve the MAX-CFL problem.

1) Greedy Algorithm: The first and most obvious algorithm that we consider is the greedy algorithm. This algorithm chooses the most critical net and, if possible, routes the net in an upper-L or lower-L fashion. If both the upper-L and lower-L routings couple with net that has already been laid out, the current net is not laid out; the most critical remaining net is then considered. The algorithm iterates until all nets have been considered.

Algorithm 1: Maximum Coupling-Free Layout Routing Greedy Heuristic

Given a set of nets N
Sort N by criticality (largest $\rightarrow$ smallest)
for each net $n \in N$
do route n in upper-L or lower-L, if possible

Theorem 7: The maximum coupling-free routing greedy heuristic takes $O(|N| \log |N|)$ time.

Proof 7: The sorting step takes $O(|N| \log |N|)$ time. The "for" loop will complete after $|N|$ iterations. Hence, we have $O(|N| \log |N|)$ run time for the algorithm. □

The greedy heuristic is a simple and fast method of finding a maximum coupling-free layout solution.

Of course, there are many shortcomings to this algorithm. First, the greedy nature of the algorithm may cause a critical net that couples with many other less critical nets to be routed. By not routing a critical net, you may be able to route a large number of other less critical nets which can lead to a better overall solution. A simple example of this situation is shown in Fig. 10. The greedy algorithm will place net A first. Then, it will place net B in an upper-L routing because it is the most critical unrouted net. Now, neither net C or net D can be placed since they both couple with net B. The best solution in terms of number of nets routed and total criticality routed is routing nets A, C, and D.

2) Implication Algorithm: We showed how to generate an implication graph from an instance of the coupling-free routing problem in Section IV-C. Now, we use some of the properties of the implication graph to create a heuristic to solve the MAX-CFL problem.

The implication algorithm tries to eliminate the bad decisions made by the greedy algorithm. It starts by determining the forcing interactions between every pair of nets. Then, it finds the nets that have a truly independent routing (either upper-L

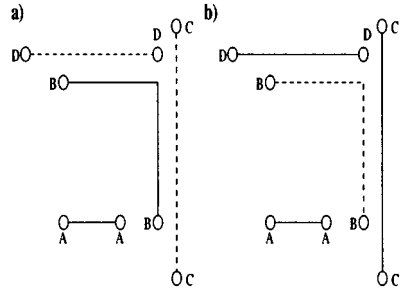

Fig. 10. An unrouted net is displayed as a dotted line; a routed net has a solid line. Assume that criticality of net $A = 100$, $B = 50$, and $C = D = 40$. (a) Greedy algorithm solution. Two nets are placed with a total criticality of 150. (b) Best solution. Three nets are placed with a total criticality of 180.

or lower-L) and routes them in the appropriate manner. An independent routing is equivalent to a route that forces no other nets (corresponding to interactions 1, 3–6 from Fig. 7). If a net only forces other nets when it is routed in a lower-L (upper-L), it will be routed in an upper-L (lower-L). The upper-L situations corresponds to interactions 3 and 5 while the lower-L situations corresponds to interactions 4 and 6. Since these routings are independent, routing these nets cannot cause a situation as described in Fig. 10. The remaining nets are routed according to a function of number of nets that they directly and indirectly force. The net with lowest value according to that function is routed first, as long it does not couple with any net that is already routed. This process continues until all of the nets have been considered.

Theorem 8: The running time of the implication algorithm is $O(|N|^3)$.

Proof 8: According to Theorem 4.3, the construction of the implication graph takes $O(|N|^2)$ time. There are $O(|N|)$ vertices in the implication graph, therefore the first "for" loop has $O(|N|)$ iterations. As stated in Theorem 5.4, the Forcings algorithm has a run time of $O(|V| + |E|)$. Note that $O(|V|) = O(|N|)$ and $O(|E|) = O(|N|^2)$. Therefore, the total run time of the "for" loop is $O(|N|^3)$. Sorting takes $O(|N| \log |N|)$ time. The final "for" loop is $O(|N|)$ time. Therefore, the algorithm requires $O(|N|^3)$ time. □

Algorithm 2: Maximum Coupling-Free Layout Routing Implication Heuristic

```
Given a set of nets N
Create an implication graph G(V, E)
R ← ∅
for each vertex v ∈ V
do r.net ← route(v)
    r.num_forcing ← Forcings(route(v))
    R ← R ∪ r
Sort R by function(direct_forcings,indirect_forcing) (smallest → largest)
for each routing r ∈ R
do if r.net is unrouted and r is routable
then route r
```

3) Maximum 2—Satisfiability Algorithm: In Section IV-B, we showed how to transform the coupling-free routing problem into an instance of 2-SAT. In this section, we show how one can use the well-known problem of maximum 2-satisfiability (MAX-2SAT) to solve MAX-CFL.

Given a set of boolean variables, U, a collection of clauses C such that each clause $c \in C$ has $|c| = 2$, and for an integer $K \leq |U|$, the *maximum 2-satisfiability (MAX-2SAT)* problem is defined as finding a truth assignment for U such that at least K clauses $\in C$ are satisfied. MAX-2SAT is NP-complete [41].

It seems that solving the MAX-2SAT problem on a transformed 2-SAT instance of CFR would be equivalent to solving MAX-CFL. Yet there are some subtle differences between them. First, the objective of MAX-2SAT maximizes the number of satisfied clauses by finding an appropriate truth assignment to the boolean variables. But, in MAX-CFL, we wish to maximize the number of routed nets; this means that we wish to minimize the number of variables in unsatisfied clauses of the equivalent MAX-2SAT instance. These are two different objective functions.

Remember that each variable corresponds to the routing of exactly one net. If a clause is unsatisfied, then the value of the two variables in that clause are not valid. For example, assume that we have two nets, A and B, that have a coupling interaction specified by the clause $(x_A \vee x_B)$.[7] If that clause is unsatisfied, it implies that x_A and x_B are both *false*, i.e., both nets A and B are routed in a lower-L pattern which causes coupling between the two nets. Therefore, we cannot route either net A or net B and still keep a coupling-free routing.

We may have a large number of unsatisfied clauses, hence we must eliminate at least one net for each unsatisfied clause. Of course, eliminating the routing of one net corresponds to removing all the forcing interactions, i.e., all the clauses where that variable exists, between that net and every other net. Therefore, the real problem becomes finding a maximum set of nets such that they are coupling-free, i.e., their 2-SAT instance is completely satisfied. This in itself is another optimization problem.

Despite these differences, a correlation between the number of satisfied clauses in the MAX-2SAT instance and the number of coupling-free routed nets exists. Therefore, we can still use a MAX-2SAT algorithm to solve the MAX-CFL problem as long as we take into account the differences. We do this by determining the number of variables in the unsatisfied clauses and removing the routing of the nets that correspond to those variables. This yields a lower bound for the MAX-CFL problem, as it is possible to remove only a subset of these nets and still maintain a valid solution.

4) Evaluation: To perform our experiments, we used five MCNC standard-cell benchmark circuits and five benchmarks from the ISPD98 benchmark suite [43] (ibm01–05). The circuits were placed into using the Dragon global and detailed placement engine [22].

Our experiments focus on reducing the added delay caused by coupling. Long nets (in terms of wirelength) have the greatest opportunity for coupling and have the largest amount of interconnect delay. Therefore, we look at the longest nets from each of these circuits. We attempt to find a coupling-free 1-D routing for the set of nets since we showed in the previous section that a set of 1-D nets will not affect the overall routability of the circuit.

First, we investigate the sensitivity of the coupling threshold. Fig. 11 shows the number of constraints when we vary the cou-

[7]This corresponds to the lower-L routing for A forcing the upper-L routing for B. See Section IV-B, interaction 3.

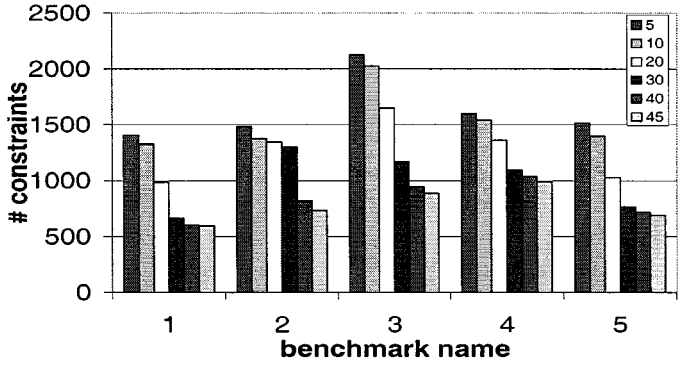

Fig. 11. The number of coupling constraints over the ISPD98 benchmark files. The coupling width over all the benchmarks is 1 unit. The coupling length varies from 5–45 units according to the legend.

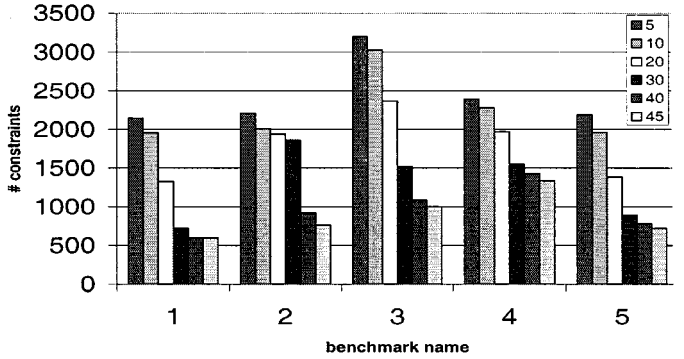

Fig. 12. The number of coupling constraints over the ISPD98 benchmark files. The coupling width over all the benchmarks is 2 unit. The coupling length varies from 5–45 units according to the legend.

pling length while the coupling width remains at 1 unit. Fig. 12 shows a similar figure when the coupling width is 2 units. Recall that two nets have a coupling interaction iff they have line segments that are at a distance of the coupling width or less and run in parallel to another for more than the coupling length. We use the ISPD98 benchmarks for comparison since they roughly have the same grid size. Furthermore, we consider the case when there are 100 nets.

We expect two general trends. First, the number of constraints should monotonically increase as the coupling length decreases. Second, the number of constraints should monotonically decrease as the coupling width increases. The rate of increase/decrease is the relevant data. It is interesting to note that the difference in the number of constraints between the two charts differs significantly when the coupling length is small (e.g., 10, 20), yet the difference is minimal when the coupling length is large (e.g., 40, 45). As the coupling length decreases, the benchmarks tend to show an exponential decrease in the number of constraints.

We compare the greedy algorithm, implication, and MAX-2SAT algorithms in terms of the number of nets routed and criticality of the nets that are routed. Net criticality is normally defined at the logic synthesis stage and is a function of the amount of slack available on a net. Unfortunately, the benchmarks do not include timing information. Hence, we need another measure of criticality. It has been shown that the delay for a wire of length l increases at the rate of $O(l^2)$ without wire sizing, $O(l\sqrt{l})$ with optimal wire sizing and linearly with

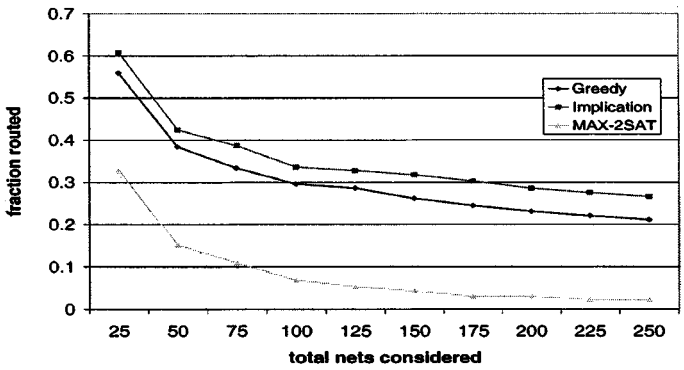

Fig. 13. Fraction of nets placed averaged over all benchmarks.

proper buffer insertion [44]. We did experiments using linear (l), l-root-l $(l\sqrt{l})$, and quadratic (l^2) functions. Of course, the criticality function can easily be changed to incorporate some other function.

To solve the MAX-2SAT problem, we used the FMSAT solver from the University of Michigan [45]. The algorithm used is similar to the Fiduccia–Mattheyses algorithm for hypergraph partitioning except that the gain update is different and there is no balance constraint. Unlike many other satisfiability solvers, FMSAT has the ability to output partially satisfied (MAX-SAT) answers when a fully satisfied answer is not achieved. In order to obtain a solution, we removed all the variables (hence nets) that are in unsatisfied clauses. Therefore, the MAX-SAT solution we obtain is a lower bound on the best possible solution generated from the solver. We could possibly obtain a better solution by removing only a subset of these nets. Yet, this is another optimization problem itself; we only wish to use the MAX-SAT solver as a comparison with the other algorithms and leave this optimization problem as potential future work.

Fig. 13 shows the fraction of nets that are placed by the greedy, implication, and MAX-2SAT algorithms. In this experiment, we used the linear function indirect_forcing $+$ $2 \times$ *direct_forcing* for the implication algorithm. We set the coupling width and length thresholds to 1 and 10, respectively.

We can see that the implication algorithm consistently finds a routing for a larger percentage of nets. Over all the experiments that we ran, the implication algorithm routes, on average, 3.38% more nets than the greedy algorithm. Both these algorithms perform much better than the MAX-2SAT solver. We believe there are several reasons for the poor performance of the MAX-2SAT algorithm. First, we are trying to maximize the number of violated variables (variables in unsatisifed clauses) which is different from the MAX-SAT objective function (maximizing the number of unviolated clauses). Also, a MAX-SAT solver is not generated specifically for MAX-2SAT. A solver that focuses on 2-SAT instances would undoubtedly perform better. Finally, as we discussed earlier, the number of violated variables is only a lower bound on the number of routable nets.

When the problem is highly constrained, the greedy and implication algorithms perform similarly. A smaller grid size and the larger number of nets adds constraints to the problem. With fewer constraints on the problem, the implication algorithm performs notably better. Table VI shows the routed net results for

TABLE VI
PERCENTAGE OF ROUTES LAID OUT FOR LARGE BENCHMARKS

Num	avqs		ibm01		ibm02	
nets	greedy	imp.	greedy	imp.	greedy	imp.
25	76%	84%	64%	72%	64%	72%
50	58%	62%	50%	74%	46%	50%
75	52%	56%	41%	46%	33%	41%
100	45%	49%	36%	37%	33%	38%
125	46%	46%	22%	29%	29%	33%

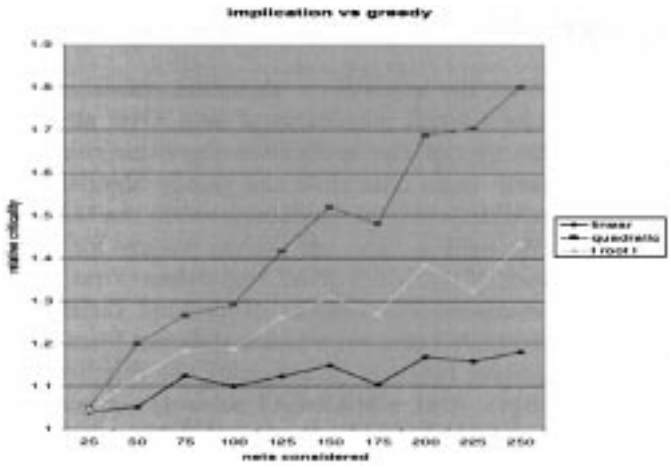

Fig. 14. Relative criticality of nets placed by the greedy algorithm compared to the implication algorithm. The results are averaged over all benchmarks. The criticality of the benchmarks are normalized to the criticality result of the implication algorithm. Therefore, a result of y indicates that the greedy algorithm laid out $y\times$ (criticality of implication algorithm).

some of the larger benchmarks. You can see that the performance of the implication algorithm is quite good on the large benchmarks, especially when we consider a small number of nets.

If we only look at the criticality of the nets routed, we see that the greedy algorithm is better than the implication algorithm. Fig. 14 confirms that the greedy algorithm outperforms the implication algorithm using a quadratic function, l-root-l, and linear function. For a linear criticality function, the greedy algorithm was approximately 1.1 times better than the implication algorithm. If we use the quadratic function, the greedy function outperforms the implication heuristic by a factor of 1.8 (when we consider the 250 most critical nets). This should be of little surprise, however, since the implication algorithm does not use the idea criticality to find a routing of the nets.

In summary, the results indicate that the implication algorithm is the best algorithm for routing the maximum number of nets. The greedy algorithm tends to find a layout with maximum criticality but performs poorly with respect to maximizing the number of nets.

V. CONCLUSION

In this work, we show show that pattern routing is a useful concept for handling coupling and increasing predictability of the routing without affecting the routability of the circuit. We argued that pattern routing is beneficial to higher level CAD tools since it allows them to choose the routings of a subset of nets while insuring the quality of the routing solution. In addition, we showed that pattern routing can help even at the global routing stage by leading the router find a better solution.

In the first part of the paper, we looked for nets that can be pattern routed without degrading the quality of the routing solution. Even with this limitation, we show that we can pattern route up to 80% of the nets. Also, we show that pattern routing works with large nets if they are γ-d routable.

In the second part of the paper, we address the issue of coupling during routing. We present algorithms and theory for a new problem named CFR which is a coupling formulation for pattern routing. We purposely define a CFR problem to be generic; this allows us to use the problem as a base algorithm to which a wide variety of extensions can be added to create more complex heuristics. We mention some possible extensions to CFR for detailed routing, single layer routing, and global routing. Additionally, we discuss an extension to the algorithm that considers the cumulative effects of coupling from multiple nets.

We show how to transform CFR to an implication graph, which takes an instance of the problem and models the dependencies or forcings that exist between the nets. We present an exact, efficient algorithm for the CFR decision problem via a transformation to the 2-satisfiability problem. The CFR decision problem will determine whether every net within a specified set is coupling-free routable.

The MAX-CFL problem is defined as finding a coupling-free routing for the maximum number of nets in a set. We show that the planar MAX-CFL problem is NP-complete. Also, we give a few heuristics for solving the general MAX-CFL problem and the greedy, implication, and MAX-2SAT algorithms.

The greedy algorithm is quite simple, yet it is an effective way of obtaining a layout with maximal criticality with small runtime complexity. The implication algorithm uses some properties associated with the implication graph to formulate a solution. The MAX-2SAT algorithm transforms the MAX-CFL problem into a 2-satisfiability instance and generates an answer using a MAX-SAT solver. Our experiments show that the implication algorithm is the best algorithm at routing the maximum number of nets; it consistently routes the largest number of nets.

ACKNOWLEDGMENT

The authors would like to thank Prof. I. Markov and A. Ramani from the University of Michigan for providing their FMSAT solver and modifying it to fit the authors' needs. Also, they would like to thank the referees for their constructive suggestions and comments.

REFERENCES

[1] M. Sarrafzadeh and C. K. Wong, *An Introduction to VLSI Physical Design*. New York: McGraw-Hill, 1996.
[2] N. Sherwani, *Algorithms for VLSI Physical Design Automation*. Boston, MA: Kluwer, 1993.
[3] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*. New York: Wiley, 1990.
[4] R. M. Karp, "Complexity of computer computations," in *Reducibility Among Combinatorial Problems*. New York: Plenum, 1972.
[5] C. Y. Lee, "An algorithm for path connection and its application," in *IRE Trans. Electronic Computer*, 1961.

[6] L. P. P. P. van Ginneken, "Buffer placement in distributed RC-tree networks for minimal elmore delay," in *Proc. Int. Symp. Circuits and Systems*, 1990.
[7] K. L. Shepard, "Design methodologies for noise in digital integrated circuits," in *Proc. ACM/IEEE Design Automation Conf.*, June 1998.
[8] D. Sylvester *et al.*, "Interconnect scaling: Signal integrity and performance in future high-speed CMOS designs," in *Proc. VLSI Symp. Technology*, 1998.
[9] H. Zhou and D. F. Wong, "An optimal algorithm for river routing with crosstalk constraints," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 1996.
[10] K. Jhang, S. Ha, and C. S. Jhon, "COP: A crosstalk optimization for gridded channel routing," *IEEE Trans. Computer-Aided Design*, Apr. 1996.
[11] T. Gao and C. L. Liu, "Minimum crosstalk switchbox routing," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 1994.
[12] T. Xue, E. S. Kuh, and D. Wang, "Post global routing crosstalk risk estimation and reduction," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 1996.
[13] K. Chaudhary, A. Onozawa, and E. S. Kuh, "Cross point assignment with global rerouting for general-architecture designs," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 1993.
[14] H. P. Tseng, L. Scheffer, and C. Sechen, "Timing and crosstalk driven area routing," in *Proc. ACM/IEEE Design Automation Conf.*, June 1998.
[15] H. Zhou and D. F. Wong, "Global routing with crosstalk constraints," in *Proc. ACM/IEEE Design Automation Conf.*, June 1998.
[16] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Coupling aware routing," in *Proc. IEEE Int. ASIC/SOC Conf.*, Sept. 2000.
[17] R. Kastner, E. Borzorgzadeh, and M. Sarrafzadeh, "Predictable routing," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 2000.
[18] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "An exact algorithm for coupling-free routing," in *Proc. Int. Symp. Physical Design*, Apr. 2001.
[19] J. Ho, G. Vijayan, and C. K. Wong, "A new approach to the rectilinear steiner tree problem," in *Proc. ACM/IEEE Design Automation Conf.*, June 1989.
[20] E. Bozorgzadeh, R. Kastner, and M. Sarrafzadeh, "Creating and exploiting flexibility in steiner trees," in *Proc. ACM/IEEE Design Automation Conf.*, June 2001.
[21] V. Vaishnavi and D. Wood, "Rectilinear line segment intersection, layered segment trees and dynamization," *J. Algorithms*, July 1982.
[22] M. Wang, X. Yang, and M. Sarrafzadeh, "DRAGON: Fast standard-cell placement for large circuits," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 2000.
[23] R. Nair, "A simple yet effective technique for global wiring," *IEEE Trans. Computer-Aided Design*, Mar. 1987.
[24] W. Dougherty and D. Thomas, "Unifying behavioral synthesis and physical design," in *Proc. ACM/IEEE Design Automation Conf.*, June 2000.
[25] D. Sylvester and K. Keutzer, "A global wiring paradigm for deep submicron design," *IEEE Trans. Computer-Aided Design*, Feb. 2000.
[26] A. Kahng, S. Mantik, and D. Stroobandt, "Requirements for models of achievable routing," in *Proc. Int. Symp. Physical Design*, Apr. 2000.
[27] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*. Boston, MA: Addison-Wesley, 1990.
[28] J. Cong *et al.*, "Performance optimization of VLSI interconnect layout," *Integration, VLSI J.*, 1996.
[29] ——, "Interconnect design for deep submicron IC's," in *Proc. IEEE Int. Conf. Computer-Aided Design*, Nov. 1997.
[30] M. Lee, A. Hill, and M. H. Darley, "Interconnect inductance effects on delay and crosstalk for long on-chip nets with fast input slew rates," in *Proc. Int. Symp. Circuits Systems*, May 1998.
[31] Y. Massoud *et al.*, "Layout techniques for minimizing on-chip interconnect self inductance," in *Proc. ACM/IEEE Design Automation Conf.*, June 1998.
[32] D. Kirkpatrick and A. Sangiovanni-Vincentelli, "Techniques for crosstalk avoidance in the physical design of high-performance digital systems," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1994.
[33] K. Kozminski, "Benchmarks for layout synthesis—Evolution and current status," in *Proc. ACM/IEEE Design Automation Conf.*, June 1991.
[34] S.-W. Hur, A. Jagannathan, and J. Lillis, "Timing driven maze routing," in *Proc. Int. Symp. Physical Design*, Apr. 1999.
[35] M. Sarrafzadeh, E. Bozorgzadeh, R. Kastner, and A. Srivastava, "Design and analysis of physical design algorithms," in *Proc. Int. Symp. Physical Design*, Apr. 2001.
[36] K. F. Liao, M. Sarrafzadeh, and C. K. Wong, "Single-layer global routing," *IEEE Trans. Computer-Aided Design*, 1994.
[37] Z.-M. Lin and Z.-W. Ro, "A heuristic planar routing algorithm for high performance single-layer layout," *manuscript*, 2000.
[38] J. Cong, J. Fang, and K.-Y. Khoo, "DUNE: A multi-layer gridless routing system with wire planning," in *Proc. Int. Symp. Physical Design*, Apr. 2000.
[39] P. Saxena and C. L. Lui, "A postprocessing algorithm for crosstalk-driven wire perturbation," *IEEE Trans. Computer-Aided Design*, June 2000.
[40] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM J. Comp.*, 1976.
[41] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W.H. Freeman, 1979.
[42] R. Raghavan, J. Cohoon, and S. Sahni, "Single bend wiring," *J. Algorithms*, June 1986.
[43] C. Alpert, "The ISPD98 circuit benchmark suite," in *Proc. Int. Symp. Physical Design*, Apr. 1998.
[44] J. Cong and D. Z. Pan, "Interconnect delay estimation models for synthesis and design planning," in *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 1999.
[45] A. Ramani and I. L. Markov, "The FMSAT satisfiability Solver: Hypergraph partitioning meets Boolean satisfiability," Univ. Michigan, CSE-TR-448-02, 2002.

Ryan Kastner (S'00) received B.S. degrees in both electrical engineering and computer engineering, in 1999, and the M.S. degree from Northwestern University, in 2000. He is currently pursuing the Ph.D. degree in the Computer Science Department at the University of California, Los Angeles, under the guidance of Prof. M. Sarrafzadeh.

He has published approximately 30 papers, including over five journal articles. His research interests include reconfigurable computing, compilers, and VLSI CAD.

Mr. Kastner received the Best TA award at UCLA for the 2001–2002 academic year.

Elaheh Bozorgzadeh (S'00) received the B.S. degree in electrical engineering with a major in electronics from Sharif University of Technology, Tehran, Iran, in 1998. She received the M.S. degree in computer engineering from Northwestern University, Chicago, IL, in 2000. She is currently pursuing the Ph.D. degree in computer science at the University of California Los Angeles.

Her research interests include physical design automation, especially routing and routability issues, FPGAs (architecture and CAD tools), and reconfigurable computing.

Ms. Bozorgzadeh is a student member of ACM.

Majid Sarrafzadeh (M'87–SM'92–F'96) received the B.S., M.S., and Ph.D. degrees in 1982, 1984, and 1987, respectively, from the University of Illinois at Urbana-Champaign in electrical and computer engineering.

He joined Northwestern University as an Assistant Professor in 1987. In 2000, he joined the Computer Science Department at University of California at Los Angeles (UCLA). His recent research interests include the area of embedded and reconfigurable computing, VLSI CAD, and design and analysis of algorithms. He has published approximately 240 papers, is a coeditor of the book *Algorithmic Aspects of VLSI Layout* (World Scientific, 1994), and coauthor of the book *An Introduction to VLSI Physical Design* (McGraw-Hill, 1996). He has collaborated with many industries in the past 15 years including IBM and Motorola and many CAD industries and was one of the main architects of Monterey Design Systems main product.

Dr. Sarrafzadeh received an NSF Engineering Initiation award, two distinguished paper awards in ICCAD, and the best paper award in DAC. He has served on the technical program committee of numerous conferences in the area of VLSI Design and CAD, including ICCAD, DAC, EDAC, ISPD, ISQED, and DesignCon. He has served as committee chairs of a number of these conferences. He is on the editorial board of the *VLSI Design Journal*, an Associate Editor of *ACM Transactions on Design Automation (TODAES)*, and an Associate Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN.