An Exact Algorithm for Coupling-Free Routing *

Ryan Kastner Elaheh Bozorgzadeh Majid Sarrafzadeh

Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90095-1596
{kastner,elib,majid}@cs.ucla.edu

ABSTRACT

In this work, we develop methods to reduce interconnect delay and noise caused by coupling. First, we explain the Coupling-Free Routing (CFR) Problem. CFR takes a set of nets and tries to find a one-bend couple-free routing for a subset of nets. A routed net must not couple with any other routed net. We define coupling as a boolean variable which is true when the coupling of two nets is greater than some threshold. Any pair-wise coupling definition can be used. We argue that this problem is useful in both global and detailed routing.

We develop an exact algorithm for the CFR decision problem via a transformation to 2-satisfiability. This algorithm runs in linear time. The decision problem determines whether the given set of nets is coupling-free routable. Next, we present the implication graph which models the dependencies associated with CFR. Also, we look at some of the properties associated with the graph.

Finally, we develop a new algorithm for the Maximum Coupling-Free Layout (MAX-CFL) problem. Given a set of nets, the MAX-CFL is defined as finding a subset of nets that are coupling-free routable. The subset should have maximum size and/or criticality. The new algorithm, called *implication algorithm*, uses properties associated with the implication graph and experiments show that it consistently finds the best solution in terms of number of nets routed as compared to previous algorithms.

1. INTRODUCTION

As fabrication technology moves into deep sub-micron (DSM) device sizes and gigahertz clock frequencies, interconnect becomes an increasingly dominant factor in performance, power, reliability and cost. In particular, 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.

In order to keep wiring resistance from increasing too quickly, many processes are scaling the wire height at a slow rate compared to wire pitch. This results in taller, thinner wires. Also, spacing between wires is decreasing in order to yield high packing densities. A detrimental side effect of these scaling trends is an increased amount of coupling capacitance. Coupling capacitance is proportional to the amount of parallel overlap between the wires and inversely proportional to the distance between the wires. In fact, coupling capacitance between wires can account for over 70% of the total wiring capacitance, even in 0.25 μm processes [3]. There are two problems introduced by coupling, delay deterioration and crosstalk.

In this work, we focus on reducing the unwanted effects caused by coupling during routing. In Section 2, we give some basic definitions, formally define coupling and introduce the Coupling-Free Routing (CFR) Problem. An exact algorithm for Coupling-Free Routing Decision Problem (CFRDP) is given in Section 3. The implication graph is presented in Section 4. An implication graph models the dependencies between nets in the CFR problem. Section 5 discusses the Maximum Coupling-Free Layout (MAX-CFL) Problem and analyzes the implication algorithm compared to the previously proposed greedy and forcing algorithms developed to solve the same problem. We conclude in Section 6.

2. PRELIMINARIES

A *multi-terminal net* n = $\{(x_1, y_1,), (x_2, y_2), ..., (x_n, y_n)\}$ is a collection of points in the plane. A *terminal* is single point of a net. A multi-terminal net can be partitioned into a collection of *two-terminal* nets (a net with exactly two points) using a number of standard techniques.

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 Figure 1. When a routing has no more than one bend, it is called a *single bend routing* [12]. We call such a net a *one-bend net*. Single bend routing provides several benefits. First, patterns can speed up the routing process. Second, pattern routing gives routing predictability pattern-routed net. Finally, it allows a more accurate estimation of interconnect resistance and capacitance [9].

The routings in Figure 1 are called the *upper-L routing* and the *lower-L routing*. 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).

We focus on routing *critical nets*. Critical nets can be defined in a variety of ways. Most often, critical nets correspond to nets that are on a critical path of a network at the logic synthesis stage. Interconnect delay of these nets should be minimal, therefore we are

*This work was partially supported by NSF under Grant #CCR-0090203

ISPD'01, April 1-4, 2001, Sibina, California, USA.

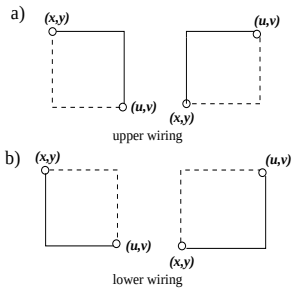

Figure 1: a) Upper-L routings b) Lower-L routings

interested in a single bend routing. A single bend routing is not only the shortest possible route between the two terminals, it also introduces only one via. Since vias further increase the wire capacitance and resistance, it is beneficial to keep them at a minimum. Also, vias negatively affect the routability of the circuit [1].

2.1 Coupling

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

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

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 d_{ij} is the distance from the center line of wire i to the center of wire j (see Figure 2).

We are trying to minimize the coupling. During routing, we can control l_{ij}, d_{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. However, this can be done as a post-processing step using a number of techniques (see [5] for a comprehensive survey and tutorial).

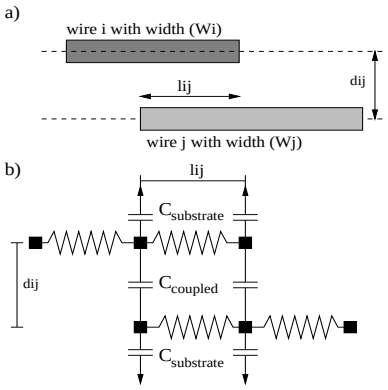

Figure 2: a) Physical coupling capacitance between two wires b) The wires modeled by resistors and capacitors

2.2 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. Any other definition of coupling between two routes can be used, as long as it is boolean e.g. there is a threshold value where they go from non-coupled to coupled. The algorithms[1] and formulations presented in this paper will remain unchanged no matter what pair-wise coupling definition is used.

For a given set of nets $S = \{N_i = \{(x_{1i}, y_{1i}), (x_{2i}, y_{2i})\} \mid 1 \leq i \leq n\}$, a (single bend) layout of S is coupling-free if there are no two routes couple. Examples of coupled and non-coupled layouts are given in Figure 3. 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) [10].

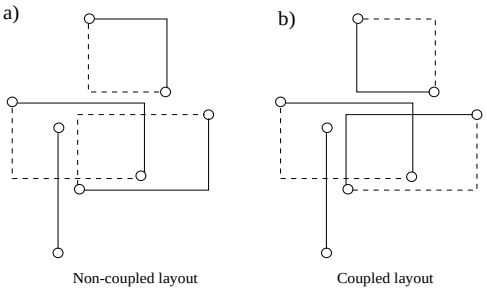

Figure 3: a) Coupling-free routings b) Non-coupling-free routings

As 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 and width of the wires on that layer. 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 dominant factor in delay of a circuit and coupling plays are large role in interconnect delay, these nets should 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, non-critical nets can be routed on the preferred layers to maximize routing resources.

Many single-layer routing algorithms have been suggested. Liao *et. al* [7] propose density routing or maze routing to perform this task. A more recent paper by Lin and Ro [13] improves on the work by Liao *et. al*. They employ a two step process. First, they find a planar set of single-bend nets. Then, they use a method based on rubber-band 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 two neighbors[2]. 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. 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

[1] The complexities of the algorithms may increase due to additional computations added by calculating the coupling.

[2] 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.

can provide a way to make large scale changes to a layout that otherwise can not 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 minimize coupling for the detailed router.

3. THE COUPLING-FREE ROUTING DECISION PROBLEM

Throughout the rest of the paper, we state theorem, lemmas, etc. without proof due to space limitation. These proofs are found in the technical report.

The Coupling-Free Routing Decision Problem (CFRDP): 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?

We solve the coupling-free routing decision problem by transforming it into an instance of the 2-satisfiability (2SAT) 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 2SAT problem can be solved in $O(|U|)$ time [11].

In order to transform an instance of CFR decision problem to 2SAT, 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 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 10 possible forcing interactions between these nets. Examples of all of the 10 cases are shown in Figure 4.

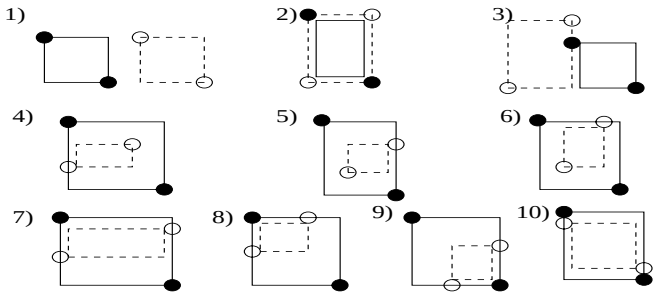

Figure 4: Examples of the 10 interactions for the coupling-free routing problem. The solid points and lines correspond to net A. The dotted lines and circles correspond to the the bounding box and terminals of net B, respectively.

The algorithm proceeds as follows:

Stage 1: Consider the $\frac{n(n-1)}{2}$ interactions (where $n = |S|$ = number of nets) between the nets. If two nets can not 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)
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 10 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 $(\overline{x_A} \vee \overline{x_B})$

The last four cases were left out for brevity; they are combination of previous cases and are easily derived.

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 $\frac{n(n-1)}{2}$ interaction relations are encoded as specified. Each of these relations becomes a clause in the 2SAT instance.

The 2SAT 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 3.1. *The coupling-free routing decision problem can be solved in $O(n^2)$ time where $n = |S|$*

4. 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.

4.1 2SAT $\propto$ Implication Graph

First, we show how an instance of 2SAT is transformable into an implication graph. In Section 3, we show how to transform an instance of the CFR problem to an instance of 2SAT. Since CFR $\propto$ 2SAT $\propto$ implication graph, CFR $\propto$ implication graph. The multi-step 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 $\Lambda = \Lambda_i(x_i \vee y_i)$ be an instance of 2SAT, where x_i, y_i are literals over $a_1, ..., a_n \in A$. We want to know when SAT(A) 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 4.1. *If there is a cycle in G containing both x and $\overline{x}$ for all x $\in$ V, A is not SAT.*

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

4.2 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) $G(V, E)$. 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 4.2. *If there is an implication $x_A \Rightarrow x_B$, there is contrapositive implication $\overline{x_B} \Rightarrow \overline{x_A}$.*

Theorem 4.3. *Given a set of nets N, the construction of the corresponding implication graph takes running time $O(|N|^2)$.*

Lemma 4.1. *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 ∈ N, then the nets N are not couple-free routable.*

Lemma 4.2. *Given a set of nets N, there is an O(|V||E|) algorithm to determine if these nets are coupling-free routable.*

For each implication case, up to two clauses are added to 2SAT in the transformation. These clauses correspond directly to edges in the implication digraph. Figure 5 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 2SAT 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 Figure 5 (c). This means that these three nets can be coupling-free routed.

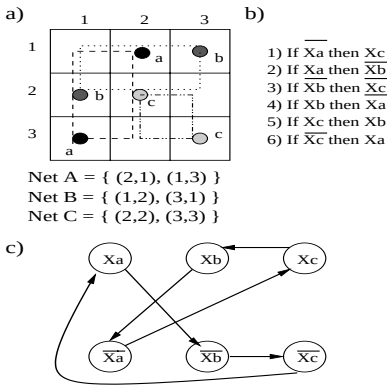

Figure 5: a) The layout of nets A, B and C. b) The implications of the nets. c) The implication graph. x_i indicates an upper-L routing of net i. The implication graph does not have any cycles containing x_i and $\overline{x_i}$, i ∈ A,B,C, therefore the nets are coupling-free routable.

4.3 Properties

Assume that we have implication graph $G(V, E)$ which is constructed from an instance of a CFR problem containing the set of nets N. 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 (u, v), then the route(u) forces route(v). This is a direct consequence of the way that the implication graph is constructed.

In an implication graph, the outdegree of v corresponds to the number of routings that route(v) forces. We call this a *direct forcing*. A routing may force a net even if it isn't a direct forcing. Referring to Figure 5, the upper-L routing of net C directly forces an upper-L routing of B. The upper-L route of B forces a lower-L route of A. So, if we choose to route the net C in an upper-L manner, then nets B and A must be laid out as upper-L and lower-L, respectively. Route upper-L of C forces two routes even though it only directly forces upper-L of B. We say that upper-L of C indirectly forces lower-L of A.

Given an implication graph $G(V, E)$ 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 number of vertices that are connected to v. A slightly modified depth-first search algorithm gives the number of forces in $O(|V| + |E|)$.

5. 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 5.1. *The Maximum Coupling-Free Layout Problem for planar layouts is NP-Complete.*

MAX-CFL can be extended to consider criticality. The criticality of a net can be defined in numerous ways. Most often, a nets 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 new heuristic to solve the MAX-CFL problem.

5.1 Implication Algorithm

We showed how to generate an implication graph from an instance of the coupling-free routing problem in Section 4. Now, we use some of the properties of the implication graph to create a heuristic to solve the MAX-CFL problem.

First, the algorithm determines the number of direct and indirect forcings of each route. Then, it sorts the routes according a function of direct forcing and indirect forcings. Finally, it tries to route the nets based on this ordering.

Theorem 5.2. *The running time of the implication algorithm is $O(|N|^3)$.*

5.2 Previous Algorithms

In this sub-section, we briefly summarize two algorithms for the MAX-CFL presented in [10], the greedy algorithm and the forcing algorithm.

The greedy 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 and the most critical remaining net is considered. The algorithm iterates until all nets have been considered.

ALGORITHM 1. *Maximum Coupling-Free Layout Routing Implication Heuristic*

Given a set of nets N
Create an implication graph $G(V, E)$
$R \leftarrow \emptyset$
for *each vertex* $v \in V$
  do *r.net* $\leftarrow$ *route(v)*
    r.num_forcing $\leftarrow$ *Forcings(route(v))*
    $R \leftarrow R \cup r$
Sort R by function(direct_forcings,indirect_forcing) (smallest $\rightarrow$ *largest)*
for *each routing* $r \in R$
  do if *r.net is unrouted and* r *is routable*
    then *route* r

The forcing algorithm tries to eliminate the the bad decisions made by greedy algorithm. It starts by determining the forcing interactions between every pair of nets. Then, it finds any nets that have a truly independent routing (either upper-L or lower-L) and routes them in the appropriate manner. An independent routing is equivalent to a route that forces no other nets. If a net only forces other nets when it is routed in a lower-L (upper-L) will be routed in an upper-L (lower-L). The remaining nets are routed according to the number of nets that they force. The net that forces the least amount of other nets 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. The forcing algorithm is a special case of the implication algorithm.

5.3 Evaluation

To perform our experiments, we used five MCNC standard-cell benchmark circuits [6] and five benchmarks from the ISPD98 suite [2]. The characteristics of the circuits are shown in Table 1. The MCNC circuits were placed into using the Dragon global and detailed placement engine [8]. The ISPD98 benchmarks are slightly modified [8] so that they could be placed by the Timberwolf placement engine. The experiments use a gridded environment simply because we have already implements a gridded router. The techniques and algorithms presented are not specific to a gridded environment. We set the coupling thresholds values at $d = 1$ and $l = 10$ i.e. two routes couple if they are closer than 1 grid space for more than 10 grid spaces.

File	Num Cells	Num Nets	Num Pins	Grid Size
MCNC benchmarks				
prim1	833	1156	3303	16x16
prim2	3014	3671	12014	32x32
avqs	21584	30038	84081	80x80
biomed	6417	7052	22253	40x40
struct	1888	1920	5407	20x16
ISPD98 benchmarks				
ibm01	12036	13056	45815	64x64
ibm05	28146	29647	127509	64x64
ibm10	68685	75940	298311	64x64
ibm15	161187	186991	716206	128x128
ibm18	210341	202192	819969	128x128

Table 1: Benchmark circuit information

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 assume that these nets will be placed on a two preferred layers – one for horizontal routings and one for vertical routings.

We compare the greedy algorithm, the forcing algorithm and the implication algorithm in terms of 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 wiresizing, $O(l\sqrt{l})$ with optimal wiresizing and linearly with proper buffer insertion [4]. We did experiments using linear (l), l-root-l ($l\sqrt{l}$), and quadratic (l^2) functions. The criticality function can easily be changed to incorporate some other function.

The implication algorithm is on par with the forcing algorithm in terms of percentage of nets routed. Table 2 shows the average percentage of routes placed by the three heuristics. The number is averaged over all the benchmarks. For this experiment, we used the linear function $indirect\_forcing + 5 \times direct\_forcing$ for the implication algorithm. Over all the experiments that we ran, the direct forcing and implication algorithm route, on average, 3.38% more nets than the greedy algorithm.

If we only look at the criticality of the nets routed, we see that the greedy algorithm is better than the direct forcing and implication algorithms. For a linear criticality function, the greedy algorithm was approximately a factor of 1.2 times better than the direct forcing algorithm. Using the same 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 direct forcing and implication heuristics by a factor of 2.54 and 1.8, respectively (when we consider the 250 most critical nets). This should be of little surprise, however, since the direct forcing and implication algorithm do not use the idea of criticality to find a routing of the nets.

In these benchmarks, we found that a majority of the nets have some sort of forcing interaction (direct or indirect) with every other net. That is, there is very little deviation in the total number of forcings over all of the nets. Therefore, implication function of (direct_forcing + indirect_forcings) is essentially random in these cases.

We varied the function in the implication algorithm. From this point forward, the notation "implication(α)" denotes that the implication algorithm used the function $indirect\_forcing + \alpha \cdot direct\_forcing$. We focused the investigation on linear functions. If $\alpha > 1$, the algorithm focuses on laying out nets with minimum number of direct forcing and the number of indirect forcings acts as secondary value. If $0 \leq \alpha < 1$, the algorithm emphasizes the number of indirect forcings; the direct forcings are secondary in this case.

Algorithm	l crit	$l\sqrt{l}$ crit	l^2 crit	% routed
greedy	1.186	1.478	1.898	30.28%
direct	1.000	1.000	1.000	35.30%
imp(1)	1.062	1.169	1.296	31.79%
imp(2)	1.078	1.116	1.156	35.45%
imp(5)	1.078	1.116	1.156	35.45%
imp(10)	1.078	1.116	1.156	35.45%
imp(.5)	1.078	1.116	1.156	35.45%
imp(∞)	1.000	1.000	1.000	31.79%

Table 2: Pecentage routed nets and criticality algorithms relative to the direct forcing algorithm. The fraction is $\frac{algorithm}{directforcing}$. Averaged over all experiments.

The results show that with the implication algorithm the criticality of the routed nets can be increased without affecting the number of nets which are routed. Admittedly, this has no apparent connection

to the algorithms as neither consider criticality data. But, by considering both direct and indirect forcings, the implication algorithm is more powerful than forcing algorithm.

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.

6. CONCLUSION

In this work, we study the Coupling-Free Routing (CFR) Problem. It addresses the issue of coupling in routing. The CFR problem considers only one-bend routings which guarantees that each net is routed with minimum wirelength, minimum number of vias and minimum delay. Additionally, one-bend routes limit the potential number of routings to two, making it easier to predict metrics such as congestion and wirelength earlier in the design flow. We purposely define CFR problem to be somewhat generic. This allows us to use the problem in a wider variety of algorithms. We argue that these problems are useful in detailed routing since they:

1. can insure that critical nets are routed with minimal coupling on a preferred layer.
2. can be incorporated into existing single layer routings algorithms, making these algorithms more useful as we go further into the DSM era.

Additionally, these problems are useful in global routing to help guide the global router to a coupling-free detailed layout.

In the future, we plan on developing better algorithms for solving the MAX-CFL problem. Specifically, we hope to incorporate criticality data into the forcing and implication algorithm to improve its performance with respect to criticality.

7. REFERENCES

[1] A. Kahng, S. Mantik and D. Stroobandt. “Requirements for Models of Achievable Routing”. In *Proc. International Symposium on Physical Design*, April 2000.

[2] C. Alpert. “The ISPD98 Circuit Benchmark Suite”. In *Proc. International Symposium on Physical Design*, April 1998.

[3] D. Sylvester *et al.* “Interconnect Scaling: Signal Integrity and Performance in Future High-speed CMOS Designs”. In *Proc. of VLSI Symposium on Technology*, 1998.

[4] J. Cong and D.Z. Pan. “Interconnect Delay Estimation Models for Synthesis and Design Planning”. In *Proc. Asia and South Pacific Design Automation Conference*, January 1999.

[5] J. Cong *et al.* “Performance Optimization of VLSI Interconnect Layout”. *Integration, the VLSI Journal*, 1996.

[6] K. Kozminski. “Benchmarks for Layout Synthesis - Evolution and Current Status”. In *Proc. ACM/IEEE Design Automation Conference*, June 1991.

[7] K.F. Liao, M. Sarrafzadeh and C.K. Wong. “Single-Layer Global Routing”. *IEEE Transactions on Computer Aided Design*, 1994.

[8] M. Wang, X. Yang and M. Sarrafzadeh. “DRAGON: Fast Standard-Cell Placement for Large Circuits”. In *Proc. IEEE International Conference on Computer Aided Design*, November 2000.

[9] R. Kastner, E. Borzorgzadeh and M. Sarrafzadeh. “Predictable Routing”. In *Proc. IEEE International Conference on Computer Aided Design*, November 2000.

[10] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh. “Coupling Aware Routing”. In *Proc. IEEE International ASIC/SOC Conference*, September 2000.

[11] S. Even, A. Itai and A. Shamir. “On the Complexity of Timetable and Multicommodity Flow Problems”. *SIAM Journal of Comp.*, 1976.

[12] V. Vaishnavi and D. Wood. “Rectilinear Line Segment Intersection, Layered Segment Trees and Dynamization”. *Journal of Algorithms*, July 1982.

[13] Z.-M. Lin and Z.-W. Ro. “A Heuristic Planar Routing Algorithm for High Performance Single-Layer Layout”. Manuscript, 2000.