Wire Packing: A Strong Formulation of Crosstalk-Aware Chip-Level Track/Layer Assignment with an Efficient Integer Programming Solution

Rony Kay
Intel Corporation
Santa Clara, California, 95052
rony.kay@intel.com

Rob A. Rutenbar
Dept. of ECE, Carnegie Mellon University
Pittsburgh, Pennsylvania, 15213
rutenbar@ece.cmu.edu

Abstract: *By focusing on chip-wide slices of the global routing grid, making a few mild geometric assumptions about layer use, and suitably abstracting pin details, we derive an extremely efficient integer linear programming (ILP) formulation for track/layer assignment. They key technical insight is to model all constraints--both geometric and crosstalk--as cliques in an appropriate conflict graph; these cliques can be extracted quickly from the interval structure of wires in a slice. We develop a "strong" linear relaxation of this problem that almost always yields the integral optimum; this solution gives us directly the maximum number of wires that can pack legally without crosstalk risk. Experiments on synthetic netlists that match statistics of wire layouts from industrial 0.25um designs demonstrate that we can pack 100 - 1000 wires optimally, or with at worst a very few overflows, in seconds.*

1. Introduction

Crosstalk noise has emerged as a significant and deleterious effect in very deep submicron technologies. Undesired voltage noise is induced through parasitic coupling capacitance from one net onto another, potentially increasing delay and decreasing signal integrity [1]. Detailed information about coupling is therefore critical during design planning. Unfortunately, coupling capacitances are a mostly local geometric effect, resulting when two wires share adjacent wiring tracks; this is known with precision only after detailed routing, very late in the design flow. Current methodologies handle this in an *ad hoc* manner, iteratively analyzing layout results and attempting repair via local re-synthesis and re-layout. It is increasingly clear that these iterations may not converge, especially when a high percentage of the nets are sensitive to crosstalk hazards [16].

The central question here is how to alter the design flow so as to avoid either overly pessimistic estimates of coupling, which cost us performance, or overly zealous structural solutions that insert many shielding wires or repowering buffers, which cost us power and area. We focus in this paper on chip-level routing, and in particular, on the useful role that a required *track and layer assignment* step can play in the overall routing flow, and a novel formulation of this problem.

Large designs are routed in a two-step process. *Global routing* plans paths on some coarse abstraction of the chip surface. In common practice, the chip is divided into a coarse grid, each of whose cells is a few tens of wire tracks on a side. For example, in a 0.25um process, one might see grids 10um on a side, which renders a $1cm^2$ chip into a 1000x1000 coarse routing grid. Both maze search and Steiner embedding techniques are used here, with the goal being control of each net's topology to manage delay and chip-wide congestion. After global routing, *detailed routing* embeds each net. Global routing provides a tight region of confinement for each detailed path, pruning the search space for detailed routing, which must deal with the geometric complexities of deep submicron design rules. Maze-style area routers, in both gridded and gridless variants, are common here.

One initial observation is that neither of these steps is the optimal place to address crosstalk. Global routing is too early: since crosstalk is a result of proximate paths after detailed routing, the best one can do here is use rough statistical estimators that discourage nets from entering regions where unwanted proximities seem likely. Conversely, detailed routing is too late: area routers that embed one net a time may encounter unresolvable ripup/reroute problems trying to embed a late-routing net that must traverse a region already dense with conflicting aggressor or victim nets.

To address these problems, we suggest a mandatory track and layer assignment step between global and detailed routing. This in itself is not new (*e.g.*, see Section II). However, we develop in this paper a simplified geometric model that preserves the wire-to-wire proximities as a target for optimization, but abstracts away the details of pin locations and low-level blockages that make detailed routing difficult. Thus, the model retains the advantages of global routing (we operate on large regions of the layout, and consider the interactions of many nets simultaneously) while exposing just enough detailed routing decisions (proximities for the global spans of individual nets) to do first-order crosstalk avoidance.

More importantly, we develop a novel integer linear programming (ILP) formulation which yields not only efficient solutions, but also tight bounds on the quality of the achievable solution. ILP solutions have a somewhat suspect history in physical design: many problems are elegantly formulated, few are practically solvable. We show how insights from polyhedral combinatorics can be used to "strengthen" a simplistic formulation, transforming a large set of loose constraints into a small set of tight constraints. This transformation allows us to obtain an efficient, practical solution.

We refer to the overall approach as *wire packing*, since the fundamental goal is to *pack* wire segments among a fixed supply of tracks and layers to avoid a set of known victim-aggressor conflicts. The remainder of the paper is organized as follows. Section II briefly surveys related work. Section III motivates and introduces our geometric model. Section IV develops our ILP formulation of wire packing. Section V presents a set of experimental results. Finally, Section VI offers concluding remarks

2. Previous Work

There have been several prior attempts at incorporating crosstalk in a routing flow. We review these briefly.

ISPD 2000, San Diego, CA.

Chaudhary [2] proposed wire spacing after detailed routing to reduce crosstalk. It is applied as a post-process and used to improve an existing layout, but it does not resolve all the violations and is not suitable for early design planning. Gao proposed a track permutation postprocessing technique to minimize crosstalk violations for channel [4] and switchbox [5] routing in one horizontal layer and one vertical layer. Similar work followed from Jhang [10]. Saxena gave an optimal perturbation algorithm in [15]. Kirkpatrick [13] proposed a crosstalk avoidance heuristic for channel routing, and developed the concept of *digital sensitivity* to decide which wires, because of the times of their likely switching, could shield other wires. Later, Vittal [19] suggested another algorithm for channel routing relying on another noise metric.

Xue [20] pointed out that crosstalk management should be tackled during global routing and before detailed routing. They proposed a heuristic to spread the crosstalk budget among the global routing cells. In each cell, wire ordering is applied independently of the other cells. However, track-wise continuity of wires that cross several cell boundaries, and segregation of incompatible nets assigned to different layers, are not supported directly. Zhou [22] suggested incorporating a greedy crosstalk-aware track assignment and layer assignment "on-the-fly" during global routing. An initial solution is constructed to minimize a global merit function. Crosstalk violations are then resolved by rip-up and re-route to minimize a sum of slacks of the constraints. Since the constraints are summed into one lumped objective function, there is no guarantee for crosstalk immunity of all nets. There is also no guarantee on the rate of convergence. It appears that it is less suitable for aggressively dense designs. However, we concur that track and layer assignment are critical components for crosstalk-aware routing.

Tseng [18] proposed an intermediate step between global and detailed routing that aims to satisfy timing constraints [18]. After a global router determines net topology in a 3D grid graph, a greedy heuristic assigns cross-points on boundaries of global routing cells, operating on one cell boundary at a time. A detailed router connects the cross-points and pins. As with [20], the heuristic does not guarantee continuity of wires as straight segments, and layer assignment is not a degree of freedom.

A common problem with all these approaches is that they tend to emphasize the constraints imposed by detailed routing inside or between small regions (notably 2-layer channels). We believe it is more productive to focus on the constraints among the *global spans of wires at full-chip level*. We introduce a model that captures this view next.

3. Geometric Model for Wire Packing

We assume a global routing decomposition that is a grid of small routing regions, but we handle these regions *slice-wise*, *i.e.*, we focus on one *entire* row or column of the chip global grid as an atomic region of geometry. This is illustrated in Figure 1. To avoid the

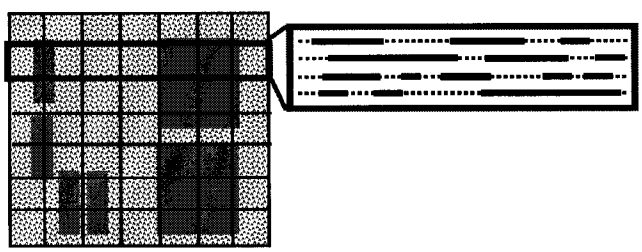

FIGURE 1. Slice-wise decomposition of wire packing attacks entire rows/columns of the global routing grid.

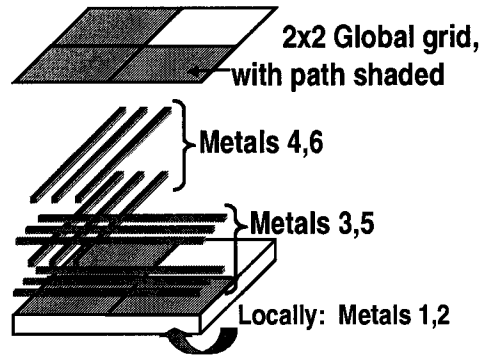

FIGURE 2. Global tracks and layers in a 2x2 global routing grid with 4 global metals. The available tracks (3) and layers (2) in one row and one column slice are highlighted.

complications of detailed routing that we described previously, we make the following simplifications:

- **Strict layer usage hierarchy:** device-level routing inside a *single* grid cell uses *only* lower-level metals, and higher-level metals are *exclusively* for chip-level interconnect. For example, grid cells could use *metal1* and *metal2*, while chip-level interconnects use *metal3* to *metal6*. We only consider the chip-level layers. In addition, we assume a rigid horizontal/vertical preference for each of the higher-level layers.
- **Track and layer model:** these chip-level interconnects are assigned *one* track and *one* layer as they cross the slice to connect *any* set of pins in different grid cells. Tracks are contiguous and parallel across an entire slice, and up all layers. Hence, routing at this level consists of straight, point-to-point connections on the layers parallel to the slice.
- **Abstracted pins:** our connections must span the grid cells containing their pins, but they do not stop "at" the pin. Rather, connections always span the *entire* width (for vertical slices, height) of any cell they touch. This is a conservative approximation since it only prohibits some legal sharing of tracks.
- **Local pin routing:** wire packing only concerns itself with the track and layer on which a signal moves from cell to cell in a slice; *all* pin-connections to these global spans are handled in lower-level metals *inside* the grid cell.

Figure 2 illustrates the geometric model. The basic idea is to abstract away details about pin locations and focus exclusively on how signals connect between grid cells. A strict HV segregation for global layers means that horizontal and vertical slices can be handled independently after global routing. We avoid complications of matching crossing points at grid boundaries, or paths to specific pin locations, choosing instead to localize these necessary decisions in each grid cell. As a practical matter, global routing cells are small enough in practice (10x10 to 20x20 tracks) that their area is that of a few gates: the number of pins in any cell is typically very small.

The underlying motivation for this model is our experience with current detailed area routers applied to chip-level routing. Such router often make suboptimal greedy path choices: a segment being routed detects a growing crosstalk threat from an adjacent line, and makes a jog to avoid further noise coupling. A few such jogs on many nets greatly complicates detailed routing. We argue

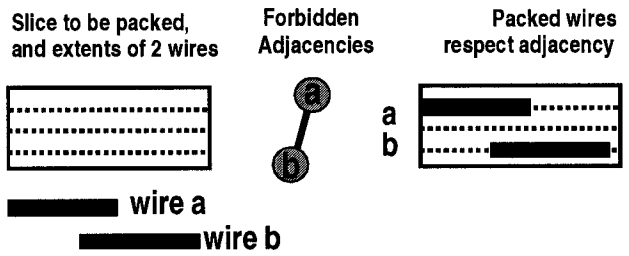

FIGURE 3. A trivial wire packing instance.

that a superior solution, in contrast, chooses *one* good global track through a slice of the global grid, at the cost of a slightly more complex path near each pin. If we assume these pin connections are done in local routing layers, and the global connections in global layers, then a wire-packed slice is a reasonable geometric approximation to the final routing. If we minimize crosstalk in the track/layer assignment, we minimize crosstalk globally.

An obvious question here is: given the limited height (10-20 tracks) of one slice, why not simply channel-route the nets? There are four reasons why a classical channel is not a useful model for us:

1. Channel routing is *detailed* routing, concerned with exact pin-to-pin connections. We want to abstract away the pin location details, to focus only on optimal track/layer assignment.
2. Our slices are chip-wide--*not* small regions--and have *fixed* height: minimizing channel density is not sufficient to solve this fixed-resources assignment problem.
3. Our goal is an *optimal* or near-optimal packing under a crosstalk risk metric, and bounds on the quality of the *optimum* solution. Channel routing is the not the right framework for this.
4. In practice, even global routing layers have obstructions, which we deal with by simply removing tracks/layers from individual grid cells in a slice. It was the difficulty in dealing with such obstructions that motivated the return to area routers.

Our view of wire packing is that it adds one more layer of *confinement* in the routing flow, from global to detailed. Global routing confines each net to some set of cells from a global grid. Wire packing further confines the global, one-directional spans of each net to a track/layer pair within these regions. However, wire packing need not be strict: detailed routing may choose to violate these track/layout assignments, but we hope it will do so only *locally*. A given span may require layer changes or small detours *near* a pin to *reach* a pin that is not exactly on the wire's assigned track/layer. A few tens of microns of such unplanned routing is negligible next to the coupling onto a chip-level wire 100, 1000, or 10,000 microns in length. Our goal in wire packing is to ensure that it is possible for the greater part of the global spans of each net segment to embed in a crosstalk-safe manner.

4. Wire Packing Via Integer Linear Programming

The input to crosstalk-aware wire-packing comprises a set of wires, a set of available tracks (per layer), and a crosstalk graph XG that determines forbidden wire adjacencies. The output is a legal assignment, such that no forbidden adjacencies and no electrical shorts exist, as illustrated in Figure 3.

Although the formulation is crisp it is computationally difficult. For example, finding a Hamiltonian path in a general graph, which is notoriously hard [7], can be transformed to crosstalk avoidance for unit length wires. Other simplified crosstalk avoidance problems in routing have been proven to be NP-Complete [20][22][13]. Here, the optimization variant is not only NP-hard, but even the approximation ratio achievable in polynomial time is in general bounded [6][21]. These negative complexity results motivate our development of a model of the problem that remains geometrically plausible (though approximate) but is practically solvable. We develop such a formulation here.

4.1 The Crosstalk Conflict Graph

The crosstalk conflict graph, XG, is a convenient abstraction of the link between the geometric and electrical aspects of wire packing. The vertices of XG correspond to wires, and the edges correspond to forbidden adjacencies (see Figure 3). We do not restrict the metric used to construct these forbidden adjacencies, we only require that they are representable as a static XG graph, given the geometry of a slice. One option is a rule-based metric, relying on precharacterization using simulations for a range of drive strengths, capacitance loads, wire separation, co-run, *etc.*

Another option is to group nets into classes (called *bins* in [13]), such that wires in the same class should not be adjacent, but there is no adjacency restriction on wires in different classes. Partitioning of nets to bins can be obtained from the results of global routing (for potential co-run signal coupling) and static timing analysis (bins represent temporal intervals in which we expect signals to switch); for discussion see Kirkpatrick [13]. We will refer to this bin model in our development to come, and in the results in Section V.

4.2 A Naive Formulation for Packing

It is straightforward to transform wire packing into a simple integer program. The decision variables of the optimization are:

$$x_{ij} = \begin{cases} 1 & \text{if wire } i \text{ is assigned to track } j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where i is an element of the set of wires W, and j is an element of the set of tracks T. In this model a wire w_i is just set of grid cells that the segment to be routed must span. Also, tracks are not restricted to be on one layer; they can be on any of the layers parallel to the slice. Each variable can be associated with a reward amount c_{ij} that measures the desirability of the assignment. We can model crosstalk-driven wire packing as finding the maximum independent set in a graph we call the *Wire-Packing-Graph* (WPG). Vertices correspond to the variables defined in (1), *i.e.*, $vertex(ij) \leftrightarrow x_{ij}$, and edges correspond to conflicting assignments. There are three types of edges (conflicts):

- Conflicts between vertices that imply that a single wire is assigned to two separate tracks.
- Conflicts between vertices that imply an electrical short: two wires overlap in the shared track to which they are assigned).
- Conflicts between vertices that imply a crosstalk violation: two wires in adjacent tracks create a known crosstalk problem.

We denote the set of edges of WPG as $E(WPG)$. Our problem can be formulated as follows:

$$\begin{aligned} &\textit{Maximize} \quad \sum_{i \in W, j \in T} c_{ij} x_{ij} \\ &\textit{Subject to} \\ &\qquad x_{ij} + x_{kl} \le 1 \qquad (ij, kl) \in E(WPG) \end{aligned} \tag{2}$$

If all the variables are restricted to values in $\{0, 1\}$, the *edge constraints* in (2) imply that only one of the two vertices incident to any edge can be in a feasible solution. Hence, all conflicting assignments are excluded. For unit weights in the objective, the optimal solution renders the objective function equal to the *maximum* number of wires that can be packed in the available tracks with no geometrical or electrical conflicts.

4.3 A Practical Formulation via a Strong LP Relaxation

Although this problem is easily expressible as a simple integer program, in particular an integer *linear* program (ILP), it remains impractical to solve. To understand why, note that most ILP solvers use a two-step solution process. First, the integrality constraints on the variables $x_{ij} \in \{0, 1\}$ are relaxed, so that now $0 \le x_{ij} \le 1$, which transforms the problem into a standard linear program (LP) that can be solved efficiently even for large problems. However, the second step is to retrieve a feasible integer solution from the relaxed solution. This is most commonly done via branch and bound methods, using the LP for bounding information, to search in the neighborhood of the relaxed solution. For this branch and bound search to terminate quickly, we need a relaxed solution that is very close to a feasible integer solution.

Unfortunately, our simple formulation of (2) yields a very difficult ILP problem. The first reason is simply the number of constraints, which is proportional to the number of edge-to-edge adjacencies in the WPG wire packing graph, which can be extremely large. Worse, the formulation is intrinsically *loose*--in the sense that the LP relaxation can be very far from any feasible integer solution. For example, the vector $[1/2, 1/2, ..., 1/2]$ is *always* a feasible solution to (2), and actually *optimal* if all c_{ij}=1 in the objective. This starting point is entirely useless to the branch and bound search. If we seek a optimal solution, we need a tighter formulation.

An appropriate set of techniques to consider here of those from *polyhedral combinatorics* [9], which studies LP formulations of combinatorial problems which are natural integer programs. The feasible domain of any LP is a *polyhedron*, defined by the intersection of the half planes associated with each linear inequality; each constraint is thus one individual facet of this polyhedron. We say an LP formulation of an IP problem is "strong" when its solution provides a better bound on, and more information about, the associated integer solution. A tighter polyhedral characterization of a combinatorial problem (*i.e.*, an LP relaxation whose solutions are intrinsically closer to an integral optimum) can be obtained by adding *strong valid inequalities* to reduce the size of the feasible region. These strong inequalities appear geometrically as *cutting planes* which slice the polyhedron and bound the solution inside to a smaller region. The real question is what the *stronger* inequalities are for a certain problem, and how to derive them efficiently.

For our problem, we have developed a suitable *strong* formulation. Although it seems natural to express the constraints in (2) in terms of the individual edges of WPG, it turns out that considering instead only the *cliques* of the WPG graph yields a much stronger set of valid inequalities, and correspondingly a much tighter LP.

Let $K(WPG)$ be the family of all the maximal cliques in WPG, and let $Q \in K(WPG)$ be any such clique in this set. Then our problem can be reformulated as:

$$\begin{aligned} &\textit{Maximize} \quad \sum_{i \in W, j \in T} c_{ij} x_{ij} \\ &\textit{Subject to} \\ &\qquad \sum_{(i,j) \in Q} x_{ij} \le 1 \quad \forall Q \in K(WPG) \end{aligned} \tag{3}$$

The basic geometrical insight here is that each clique collapses into a single, stronger constraint the many individual loose constraints implied by each of the individual conflict edges in $E(WPG)$ from (2). Among all possible cliques, we need only consider *maximal* cliques since they imply all other clique constraints (and, indeed, all individual edge-constraints in (2)). More formally, the edge-formulation and clique-formulation are equivalent if $x \in \{0, 1\}$, however, they are *not* equivalent if integrality is *relaxed*. For example, revisiting the vector whose elements are all $1/2$, maximally distant from any integer solution, we find it is no longer feasible for the clique formulation.

When the variables are restricted to $\{0, 1\}$, we call (2) the *Edge Integer Program* (EIP) and we call (3) the *cliQue Integer Program* (QIP). When the variables are relaxed to continuous values, we call (2) the *Edge Linear Program* (ELP) and (3) the *cliQue Linear Program* (QLP). Our focus is now on the properties of a QIP/QLP solution strategy. To be practical, we must show the following:

1. That we can efficiently *find* the maximal cliques which become the constraints in the QIP and QLP programs.
2. That the *number* of these maximal cliques is not only tractable, but also in practice much smaller than the number of edge constraints necessary in the naive EIP/ELP program.
3. That the relaxation from QIP to QLP is, in fact, a *stronger* relaxation, yielding both a superior value for the objective common to both (2) and (3), and a solution closer to integral.

Note that the number of maximal cliques can be exponential in the size of a graph--for a general graph [7]. However, a critical motivation behind our geometric model for a slice is that the wires have an *interval* structure that can be associated with an underlying *interval graph* [8]. We call maximal subsets of wires that overlap a common coordinate (*e.g.*, an x coordinate in a horizontal slice) *maximal-intersecting-subsets*, and denote them S_I. In an interval graph, these are the maximal cliques, and they can be found in linear time via a one-dimensional sweep down the slice; see [8].

To determine the number of these cliques, we need only perform a careful case analysis of the *types* of geometric constraints that these maximal cliques imply. Figure 4 illustrates the cases. There are 3 relevant classes of cliques, mirroring the analysis from the previous section:

- **Type I Clique Constraints:** prevent a single wire from being assigned to multiple track. They create constraints of the form:

$$\sum_{j \in T} x_{ij} \le 1 \qquad \forall i \in W \tag{4}$$

- **Type II Clique Constraints:** prevent overlapping wires from being assigned to the same track. Denote by $\widehat{F}$ the family of all

maximal intersecting subsets of wires S_I; then these constraints can be derived directly from these subsets, and must be duplicated for each track $j \in T$:

$$\sum_{i \in S_I} x_{ij} \leq 1 \qquad \forall (j \in T) \quad and \quad \forall (S_I \in \widehat{F}) \qquad (5)$$

- **Type III Clique Constraints:** forbid wires with a known crosstalk conflict (from the crosstalk conflict graph XG) from being assigned to adjacent tracks. Type III constraints are rather more subtle to derive. Let $XG(S_I)$ be the subgraph induced in XG by an overlapping subset of wires S_I. It is not difficult to enumerate the maximal cliques of this induced subgraph because its size is bounded by the number of tracks in the slice (usually $|T| \leq 20$). The fundamental observation is that no two wires in these cliques can be assigned to adjacent tracks without conflicts (see again Figure 4). We denote by A_s the sets obtained from the following Cartesian product,

$$A_s = \{max\text{-}clique\text{-}of\ XG(S_I)\} \times \{t_1, t_2\}$$

where $t_1, t_2 \in T$ are any *adjacent* tracks in the slice. Each set A_s indexes all the ways in which a maximal set of *mutually* crosstalk-interacting nets might be *illegally* placed on adjacent tracks in the slice. Note that there are several sets A_s for each S_I that imply a Type III constraint. The constraints are then:

$$\sum_{(i,j) \in A_s} x_{ij} \leq 1 \quad \forall A_s \ where\ (\exists S_I\ and\ \exists t_1, t_2\ \ adjacent\ tracks) \qquad (6)$$

If our crosstalk model is based on a partitioning of nets into temporal bins as in [13], we can use the interval properties of the slice to find the sets A_s in strict linear time.

A straightforward case analysis suffices to show that the type I, II, III cliques specified above are all *maximal* in the WPG. We could continue to derive "higher-order" constraints similar to the Type III constraints, each involving more complex adjacencies, for example, forbidden triples of adjacent tracks. Luckily, this is unnecessary: the edge constraints in the naive EIP formulation are implied by the set of cliques of types I, II, III alone. The proof involves a case analysis that shows that these cliques form an *edge clique-cover* of the wire packing graph, and hence the original constraints associated with each edge from *WPG* are all captured.

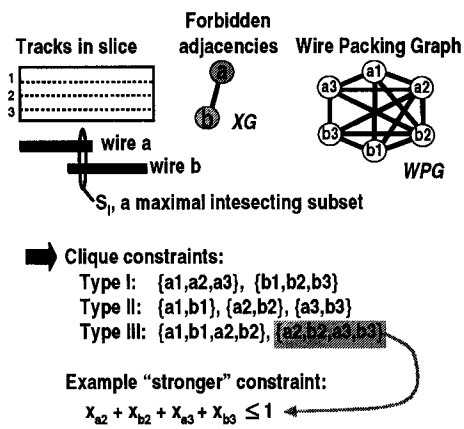

FIGURE 4. Trivial wire packing instance and its associated wire packing graph, showing examples of the 3 types of maximal cliques.

Next, we need to show that the *number* of cliques, and hence the number of constraints, is manageable. Note that the size of the maximal intersecting subsets of wires $S_I \in \widehat{F}$ is bounded by the number of tracks $|T|$ in a slice, which is typically less than 20. Therefore also the size of the cliques in each induced subgraph $XG(S_I)$ is similarly bounded by $|T|$. Furthermore, the number of maximal intersecting subsets is smaller than the number of wires $|W|$. Given these facts, a simple enumeration over the indexing sets of the constraints in (4), (5), (6) (*i.e.*, combinations of tracks, cliques, wires) shows that there are $O(|W|)$ constraints of Type I, and $O(|W||T|)$ constraints of Type II and of Type III. Hence, as a practical result, a slice with ~1000 wires and 10-20 tracks generates a very manageable number of constraints for current LP solvers.

Finally, we can show [12] that the relationship between all these programs is as follows.

$$max\{EIP\} = max\{QIP\} \leq max\{QLP\} \leq max\{ELP\} \qquad (7)$$

So, rather than focus on the naive formulation of EIP and its generally useless relaxation to ELP, we focus instead on the QIP integer program, and its tighter relaxation to QLP.

Using these clique constraints, we can finally rewrite our optimization problem in its tightest form:

$$\begin{aligned} &Maximize && \sum_{i \in W, j \in T} c_{ij} x_{ij} \\ &Subject\ to && \\ & && Q\bar{x} \leq \bar{1} \\ & && \bar{0} \leq \bar{x} \end{aligned} \qquad (8)$$

where Q is a *clique-vertex incidence matrix*, a zero-one matrix where each row corresponds to a clique (from (4), (5), (6)) and each column to a variable $x_{ij} \in \bar{x}$. The relaxation to the QLP problem has $O(|W||T|)$ variables and $O(|W||T|)$ constraints, and can be formulated in *linear* time. (For additional details, see [12].) As we shall demonstrate experimentally in the subsequent section, this formulation is not only efficient, but *remarkably* tight.

4.4 Retrieving an Integer Solution from QLP

Branch and bound methods are only one option for converting the relaxed fractional solution of QLP (8) into a real integer solution. In previous work on zero-one integer programming, Raghavan and Thompson developed the technique of *probabilistic randomized rounding* [14]. The basic idea is to interpret the fractional solution of an LP relaxation as a set of *probabilities*, and round the variables randomly to zero or one as *independent* Bernoulli-trials. The technique is attractive in our application both for its speed (it involves no search), and for the practical reason that a *slightly* infeasible integral solution, rapidly obtained, is valuable in itself. In this context, slight infeasibility amounts to a few wire overflows that cannot be embedded without geometric or crosstalk conflicts. This is consistent with the behavior of current global routers, which often overflow a handful of wires and simply inform the detailed router of the problem. For us, these few wires should be removed and then

```
Function Constructive_Randomized_Rounding
Input:  W - ordered set of wires sorted lexicographically
        T - set of available tracks, potentially on a few layers
        XG - forbidden adjacencies
        F - fractional solution of QLP (probabilities)
Output: A - legal assignments {(w,t): w ∈ W, t ∈ T, no conflicts}
        overflow - number of wires that were not assigned
Init:   overflow = 0, A = ∅

foreach w ∈ W in order
        flag = false
        while not all tracks were attempted and rejected for w
                t = random_next_track(T, F)
                if A ∪ (w, t) is legal
                        A = A ∪ (w, t), flag = true
                        break the while loop
        if flag = false then overflow = overflow + 1
end
```

FIGURE 5. Outline of Constructive Randomized Rounding.

rerouted globally--they *cannot* pack locally in this slice.

However, the result of a pure randomization for packing yields an infeasible solution with high probability. Therefore we propose a *constructive randomized rounding* that performs *dependent* random trials in a given deterministic order. Our experiments confirm that the structured approach is superior to pure randomization in retrieving an optimal or high-quality, nearly-feasible solution.

The constructive randomized rounding algorithm appears in Figure 5. The set of wires W is sorted lexicographically, by the left (or bottom) endpoint and length, in linear time using bucket sort. The function *random_next_track* selects randomly a track that has not yet been rejected for the active wire w, which takes $O(1)$ time. Attempts are made to find a feasible assignment for w until one (with respect to the current partial assignment A) is found, or all the tracks were rejected. The order of attempts to assign w to specific tracks is *biased* with probabilities according to the fractional solution of QLP (8).

The mechanics of this random biasing are worth noting. We interpret the fractional solution x_{ij} from QLP as the probability we should assign wire $w_i \in W$ to track $t_j \in T$. However, we must account for the case when the QLP solution indicates that *not* all wires can be packed. To do this, we also allow wire w_i to *opt out* of assignment, to become an *overflow*, with this probability:

$$Pr(w_i \text{ is overflow}) = 1 - \sum_{\forall (j \in T)} x_{ij} : \qquad (9)$$

If the QLP solution indicates that all wires can be packed, then this overflow assignment probability is zero.

This overall procedure is itself repeated until the optimum is found or a limit on the number of iterations is reached. Ultimately, the configuration with fewest overflows is used. In practice, this process takes negligible time, *e.g.*, milliseconds of CPU time per randomized trial.

5. Experimental Results

We describe a set of experiments undertaken to explore how well our ILP-based wire packing formulation of track/layer assignment works across a range of benchmarks. We describe our benchmarking strategy, and a set of empirical results in this section.

FIGURE 6. Simple wire-packed example: 25 wires in 6 tracks with 3 crosstalk conflict classes.

5.1 A Simple Example

As an introductory illustration, Figure 6 shows a small example with 25 wires packed into 6 tracks on 1 layer. Crosstalk conflicts occur between like-shaded wires; note that no like-shaded wires occupy adjacent tracks. Geometric density is 5 here: in the absence of the crosstalk constraints, this simple example only requires 5 tracks.

5.2 Benchmarking Strategy

The effort required to fracture an industrial routing flow, disentangle tightly integrated global and detailed routing tools and insert a wholly new step, is significant. Unfortunately, most public (mainly academic) tools are also unsuitable candidates due to their channel-centric, few-routing-layers limitations, and the small scale of public benchmarks. Hence, to begin to explore the limits of wire packing on realistic designs, we used a "bottom-up" approach, and synthesized a variety of benchmark netlists based on statistical models that started with *wire sampling* of large industrial designs.

First, we extracted "slice-shaped" regions of upper-level metals from several real 0.25um industrial ICs from IBM. (Extraction was *ad hoc* per specific designs and formats, and accomplished via a set of custom translators we created at IBM). Unfortunately, this detailed routing data is not sufficient for immediate use in our experiments, because (1) the data is not coarse-grid global routing, and (2) the data lacks proprietary structural, timing, or crosstalk annotations.

Therefore we used these slice-samples as *seeds* for a set of statistical generators. We found the left-to-right spans of the nets in the slice and normalized the coordinates to integer units in a given range (usually 0-500 or 0-1000 units). Then we removed the nets below a certain span threshold (1-2% of slice width), because we assume the short local wires are *intra*-grid-cell, on local layers, not treated by wire packing. We then fitted statistical distributions, via least squares methods, to generate a correctly distributed random *length* (span) and random *center-point* for each wire. A technical point worth noting: since most global routers do not perform layer assignment, we do not give special treatment to the number of layers available. So, for example, if a slice has a height that allows 10 tracks on *metal4* and 5 (*wider*) tracks on *metal6*, we consider the capacity of the slice to be 15 tracks, even though the nets can be assigned to *either* of the 2 layers. Example distributions appear in Figure 8; details appear in [12]. We focused on the distributions shown at right in the figure, which were most common for large

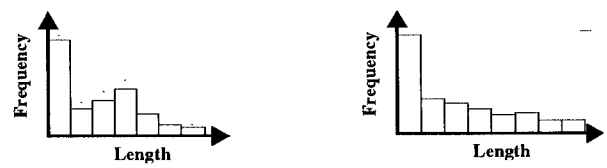

FIGURE 7. Common wire length distributions for slices: through datapaths (left), through random logic blocks (right)

slices. We generated netlists with density roughly 10-20 tracks.

For crosstalk susceptibility, we used timing bins as in [13], and randomly allocated nets to bins. Recall that these can be thought of as modeling small time intervals in a clock cycle: nets in the same bin can switch simultaneously, and so if they have enough potential co-run, they pose a crosstalk conflict. More bins implies a finer discretization of when simultaneous switching can occur. We used 3 or 4 time bins, which is consistent with current design practice.

5.3 Results from QLP/QIP Attack

Table 1 shows results from 8 benchmarks with between 100 and 530 wires, geometric density 10 to 20 tracks, physical slice capacity 11 to 23 tracks, 1 or 2 global layers, 3 or 4 timing bins. The table shows benchmark details, wire packing results (QLP/QIP), the size of constraint sets for the naive ELP/EIP formulation versus QLP/QIP, and the results for both the LP solve for each QLP problem, and the subsequently retrieved integer solutions.

Perhaps the most striking feature of this data is the tightness of the QLP bound: *for almost all problems, the LP relaxation yields directly the optimum value of the integral solution.* Note also the physical interpretation of this optimum value: for unit per-wire rewards in the objective of (8), *this value is the maximum number of wires that can be packed legally in the slice without geometric or crosstalk conflict.* This is a surprisingly practical result from an LP relaxation; we can estimate after just the LP solve how many overflows cannot embed in the slice. As we claimed earlier, this is a remarkable strengthening of the naive formulation of the ILP. It is also one with vastly fewer constraints, as can be seen in the table.

The LP solutions we must solve for QLP are also quick to obtain. Note that two outcomes are possible. If a solution is *feasible*, we expect a packing bound as in column 11 of the table. But if the solution is *infeasible*--we *cannot* pack all these wires in this slice--then for these experiments we increase by one the track capacity of the slice and try again. Of course, real slices are *fixed* height; here our goal is to understand the limits of the formulation. For example, our benchmark with 100 wires, from which 97 can be packed into 11 tracks, is solved to optimality in 6 seconds on a 300MHz IBM workstation using CPLEX 4.0. The benchmark with 530 wires in 4 bins is feasible in 23 tracks but not all the wires can be packed in 22 tracks. Infeasibility of packing in 22 tracks is proven in 15 seconds.

One iteration of constructive randomized rounding takes about 3 milliseconds. Hence, it is fast to do 1000s of trials and use the best integer solution--possibly with a few wires opting out as overflows--as our final packing. This is faster and more predictable than typical branch and bound solutions, and well-suited to the early design-planning uses we think are appropriate targets for the wire packing model.

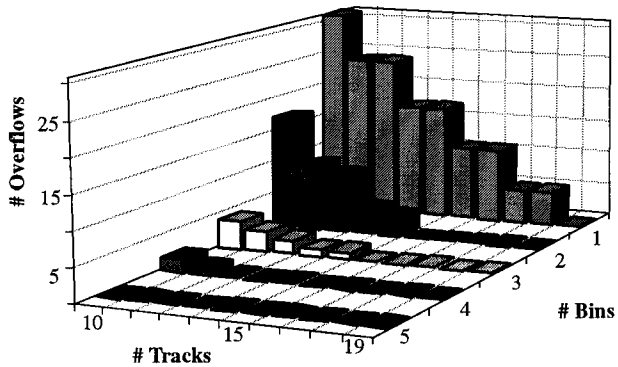

FIGURE 8. Analysis of overflows versus tracks and bins.

Next, consider a sensitivity experiment in the style of Kirkpatrick's results from [13]. Figure 8 analyzes an instance of 100 wires with density of 10 tracks for a range of crosstalk timing bins. One single bin is, of course, unreasonably pessimistic, since it assumes *all* wires create mutual crosstalk conflicts. However, as we divide the clock cycle into more intervals and partition simultaneous switching events over more bins, we see that wire packing can be very effective in embedding the wires with few overflows.

To verify that the behavior exhibited in Figure 8 is not an anomaly of a specific example, we experimented with many other test cases as well. For example, consider a netlist of 600 wires with geometric density 10 tracks that needs to be packed in 12 tracks. Table 2 shows the packing and overflow behavior as we increase from 1 to 6 timing bins. As another example, we varied the number of available tracks in the slice from 10-15 for a netlist with 700 wires, geometric density 10 tracks, partitioned into 3 timing bins. We observed the same qualitative behavior, as shown in Table 3.

The important observation across all these benchmarks is that with even a fairly crude 3-bin resolution of simultaneous switching, the wire packing approach can embed all nets, or all but a handful, in a crosstalk-safe manner at a typical cost of only a very few tracks beyond the basic geometric density of the wire intervals in the slice. We regard this as a very satisfactory result for our initial implementation of these ideas.

Benchmark Netlists: Details					Packing Results		Constraint Set Sizes			Tightness of IP to LP Relaxation	
#Wires	#Tracks/ #layers	#Time bins	Density (#tracks)	Vars (# x_{ij})	Packing (#tracks)	Overflows (#wires)	ELP	QLP: Types I/II/III	QLP: Total	Fractional QLP Bound	QIP Integral Optimum
100	11 / 1	3	10	1100	11	3	35808	100/693/580	1373	97.83	97
100	15 / 2	3	10	1500	15	0	50624	100/945/812	1857	100	100
195	15 / 1	3	10	2925	15	0	109910	195/2085/1708	3988	195	195
195	13 / 1	3	10	2535	13	1	94805	195/1807/1404	3406	194	194
314	18 / 2	4	12	5652	18	0	268991	314/4374/3400	8088	314	314
314	16 / 2	4	12	5024	16	1	238805	314/3888/3000	7202	313	313
530	23 / 2	4	20	12190	23	0	839621	530/7107/6820	14457	530	530

TABLE 1. Wire packing results: Benchmarks, packing, constraint sets, tightness of relaxation

Measured	Number of Timing Bins					
	1	2	3	4	5	6
Packing (max #wires)	473	574	592	596	600	600
% Overflows	21.1%	4.3%	1.3%	0.7%	0%	0%

TABLE 2. Varying number of bins for 600 wires in 12 tracks

Measured	Number of Tracks					
	10	11	12	13	14	15
Packing (max #wires)	677	696	696	700	700	700
% Overflows	3.3%	0.6%	0.6%	0%	0%	0%

TABLE 3. Varying number of tracks for 700 wires in 3 bins

5.4 QLP Implementation Issues and Optimizations

There are several optimizations relevant to the LP solve itself that can be exploited for speedups; [12] has details. The most notable is that we found it useful to use the *dual simplex algorithm*, which is equivalent to applying the simplex LP algorithm to the dual problem. If the original (called *primal*) form of the problem has n variables and m constraints, the dual has m variables and n constraints. An optimal solution of the dual problem implies an optimal solution of the primal problem, and vice versa. Common wisdom is that the runtime of the simplex algorithm is more sensitive to the number of constraints than the number of variables. Since the QLP formulation typically has more constraints than variables, the dual is usually faster to solve.

In practice, we also use the dual form to accelerate the check for infeasibility in the LP. Recall that the optimum of the QLP problem bounds the number of wires that can be packed in a slice with no geometric or crosstalk violations. To prove that packing *more* than a certain number of wires, say R wires, is infeasible, we solve the dual form and monitor the progress of the objective value toward the optimum, as the LP solver moves around the facets of the polytope. Since this objective is monotonically *decreasing* for the dual LP, we can terminate when (if) this value drops below R.

Another critical shortcut worth note is that we can actually *omit* one of the three classes of crosstalk constraints (cliques) from the QLP formulation to obtain a much faster solution, at the cost of a small reduction in the tightness of the relaxation. Omitting the Type II constraints of equation (5), which prevent overlapping wires from being assigned the same track, results in a surprisingly negligible increase in the LP bound, typically less than 0.1%. Indeed, this further relaxation usually gives the *same* result as a solution with all three constraint sets. The reason is that the Type III constraints *almost* cover the same set of geometric conflicts. However, the reduction in the number of constraints, and the lesser degeneracy of the smaller model, can give significant speedups.

The final optimization is to use the constructive random rounding algorithm of Figure 5 to actually *generate* an approximate solution *before* starting the LP, and use it to accelerate the LP by starting at a superior basic feasible solution (*i.e.*, use it as a so-called "hot start" for the LP). The key idea is to replace the QLP-derived assignment probabilities in the algorithm with *uniform* probabilities. The resulting solution is inferior to a real QLP solution, but fast to compute, and quite useful in this context.

Combining all these ideas, we have observed speedups of 10 to 100X over an unoptimized LP formulation. This renders problems with 100 to 1000 wires packable in typically 1 to 10 seconds. The quality of results also holds for larger netlists. For example, a slice with 1021 wires with geometric density 14 tracks, with 2 layers and 4 timing bins, packs into 17 tracks with only 4 overflows.

6. Conclusion

By focusing on chip-wide slices of the global routing grid, making a few mild geometric assumptions about layer use, and abstracting away pin details, we can derive an extremely efficient ILP formulation for track/layer assignment. They key insight is to model both geometric and crosstalk conflicts as cliques in an appropriate conflict graph; these can be extracted quickly from the interval structure of the slice. The LP relaxation of this problem almost always yields the integral optimum solution, which gives directly the maximum number of wires that can pack legally without crosstalk risk. Experiments on synthetic netlists that match statistics of wire distributions in industrial 0.25um designs demonstrate that we can pack 100 - 1000 wires optimally, or with at worst a very few overflows, in seconds. These preliminary results argue for further consideration of how wire packing might integrate into a complete routing flow.

Acknowledgment

We thank IBM and Paul Bassett, John Cohn, Phil Honsinger and Jose Neves in particular for their support, and access to routing data. This work was funded by DARPA, NSF (contract 9420372), and IBM.

References

[1] H. B. Bakoglu, *Circuits Interconnections and Packaging for VLSI*, Addison-Wesley, 1990

[2] K. Chaudhary, A. Onozawa, E.S. Kuh, "A Spacing Algorithm for Performance and Cross-talk Reduction", *Proc. DAC.*, 1993

[3] J. A. Davis, V. K. De, J. D. Meindel, "A Stochastic Wire-Length Distribution for Gigascale Integration", *IEEE TED*, Mar 1998

[4] T. Gao and C. L. Liu, "Minimum Crosstalk Channel Routing", *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design*, 1993

[5] T. Gao and C. L. Liu, "Minimum Crosstalk Switchbox Routing", *Proc. IEEE/ACM Int'l. Conf. on Computer-Aided Design*, 1994

[6] U. Feige, et al., "Approximating Cliques is almost NP-Complete", *Proc. IEEE Symp. on Foundation of Computer Science*, 1991.

[7] M. R. Garey, D. S. Johnson, *Computers and Intractability - a Guide to the Theory of NP-Completeness*, Freeman & Co., 1979

[8] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, 1980

[9] R. L. Graham, M. Grotschel, L. Lovasz (Eds), *Handbook of Combinatorics*, Elsevier, 1995

[10] K. S. Jhang, S. Ha, C. S. John, "COP: A Crosstalk Optimizer for Gridded Channel Routing", *IEEE Trans. on CAD*, Apr 1996

[11] R. M. Karp, "An Introduction to Randomized Algorithms", *Discrete Applied Math.*, 34, pp165-201, 1991

[12] R. Kay, *Algorithmic Approach to Design and Optimization of VLSI Interconnect*, Ph.D. Thesis, Dept. of ECE, Carnegie Mellon Univ., Aug. 1999.

[13] D. A. Kirkpatrick, A. L. Sangiovanni-Vincentelli, "Techniques for Crosstalk Avoidance in the Physical Design of High-Performance Digital Systems", *Proc. ICCAD* 1994

[14] P. Raghavan, C. D. Thompson, "Randomized Rounding: A Technique for Provably Good Algorithms and Algorithmic Proofs", *Combinatorica*, 7(4), 1987.

[15] P. Saxena, C.L. Liu, "Crosstalk Minimization Using Wire Perturbation," *Proc. ACM/IEEE DAC*, June 1999.

[16] Semiconductor Industry Association, "The National Technology Roadmap for Semiconductors", 1997

[17] D. Sylvester, K. Keutzer, "Getting to the Bottom of Deep Submicron", *Proc. Int'l Conf. on Computer-Aided Design*, 1995

[18] H. P. Tseng, L. Scheffer, C. Sechen, "Timing and Crosstalk Driven Area Routing", *Proc. IEEE/ACM Design Automation Conf.*, 1998

[19] A. Vittal, M. Marek-Sadowska, "Crosstalk Reduction for VLSI", *IEEE Trans. on Computer-Aided Design*, Vol. 16, No. 3, Mar 1997

[20] T. Xue, E. S. Kuh, D. Wang, "Post Global Routing Crosstalk Synthesis", *IEEE Trans. on Computer-Aided Design*, Dec. 1997

[21] M. Yannakakis, "Expressing Combinatorial Optimization Problems by Linear Programs", *J. of Computer and System Sci.*, 43, 1991.

[22] H. Zhou, D. F. Wong, "Global Routing with Crosstalk Constraints", *Proc. IEEE/ACM Design Automation Conf.*, 1998.