Faster and More Accurate Wiring Evaluation in Interconnect-Centric Floorplanning

Hung-Ming Chen¹, D.F. Wong¹, Wai-Kei Mak², and Hannah H. Yang³
¹ Department of Computer Science, University of Texas at Austin, Austin, TX 78712
² Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620
³ Strategic CAD Labs, Intel Corporation, Hillsboro, OR 97124

ABSTRACT
In deep submicron (DSM) era, the communication between different components is increasing significantly. It is not uncommon to see floorplanning problems with a relatively small number of blocks (e.g., 50) but has a large number of nets (e.g., 20K). Since existing floorplanning algorithms use simulated annealing which needs to examine a large number of floorplans, the increasing number of nets has made interconnect-centric floorplanning computationally very expensive. Moreover, there is almost no systematic way to resolve the congestion problem in such magnitude of number of nets in a given floorplan. In this paper, we present a simple yet effective idea to significantly reduce the runtime of interconnect-centric floorplanning algorithms. Our idea is to group common nets between two blocks into a single net. This faster wiring evaluation technique is very effective. We also present a more accurate global router for wiring evaluation based on Lagrangian Relaxation. The new router helps further congestion reduction while doing interconnect planning in floorplanning. We have incorporated our algorithms into [2] and observed dramatic improvement in runtime. For a 33-block 15K-net problem, we reduced runtime from over 23 hours to less than 50 minutes.

1. INTRODUCTION
With VLSI technology entering the deep submicron (DSM) era, devices are scaled down to smaller sizes and placed at an ever increasing proximity. Meanwhile, with the increase of die sizes, more functions are integrated into one chip. All these significantly increase the communication between different elements, thus increasing the amount of interconnect on a chip. Furthermore, the scaling down of fabrication geometry also makes interconnect delay a dominant factor in total circuit delay [3].

In physical design, the netlist is generated after the partitioning step while the circuits are restructured and functional blocks are obtained. The subsequent floorplanning, placement, and routing steps use the netlist to perform various kinds of optimizations to meet certain constraints. Due to the interconnect-dominant factor in modern VLSI design, communication between components is increasing dramatically. Most of logic hierarchy are flattened during placement and routing, so it is very possible that there are hundreds of thousands number of nets, while there are only tens of functional blocks in contrast. It goes without saying that the complexity of evaluating such netlists with enormous size of nets is extremely high. How to model and evaluate those high-density interconnection is becoming one of the most challenging issues in modern high-performance VLSI design.

Very recently, interconnect planning (especially global interconnects) is the key issue in modern VLSI physical design [10, 9]. However, with tens or hundreds of thousands of interconnect in standard-cell or full-custom design after circuits partitioning, interconnect planning is time-consuming. Many floorplanning algorithms, including slicing and nonslicing, have been proposed in past decades [11, 14, 2, 7, 8, 5], but they did not take interconnect planning into account except [2]. In [2], Chen et al. integrate floorplanning and efficient interconnect planning. Since every wiring evaluation step uses a simple-geometry global router, [2] can only process moderate-sized netlists (e.g. 1000 nets) but will not be efficient to solve problems with very large netlists (e.g. 10K nets). Furthermore, [2] used a heuristic global router to sequentially route global interconnects, the performance of which depends on the ordering of nets to be routed.

Since existing floorplanning algorithms use simulated annealing which needs to examine a large number of floorplans, the increasing number of nets has made interconnect-centric floorplanning computationally very expensive. In this paper, we present a simple yet effective idea to significantly reduce the runtime of interconnect-centric floorplanning algorithms. Our approach is to group common nets between two blocks into a single net. Before grouping the nets, multi-terminal nets are first decomposed into two-terminal nets. Bounded-degree hypergraph-to-graph transformation is used to preserve the constraint for pin-limit in a block. This net reduction technique is very effective. Suppose we are given a problem with 50 blocks and 20K nets. After net reduction, we can have at most 1250 nets (which is the case

*This work was partially supported by the National Science Foundation under grant CCR-9912396, and by grants from Intel, IBM and Avant.


The rest of the paper is organized as follows. Section 2 describes the wiring evaluation. Faster wiring evaluation and the algorithm for net reduction are presented in Section 3; more accurate wiring evaluation and Lagrangian relaxation technique application are presented in Section 4. Experimental results are shown and the concluding remarks are presented in Section 5.

2. WIRING EVALUATION IN INTERCONNECT CENTRIC FLOORPLANNING

In physical design, the netlist is generated after partitioning step while the circuits are restructured and functional blocks are obtained. The subsequent floorplanning, placement, and routing steps use the netlist to perform various kinds of optimizations to meet certain constraints. In the literature of floorplanning research, the focus of optimization was to minimize the total packing area as well as the wiring cost. In modern VLSI design, the size of design becomes much larger, thus the complexity of netlist is extremely higher than ever. Since the interconnect-centric floorplanning tool plays more important role in physical design, how to model and evaluate large number of interconnections has become one of the most challenging issues in modern high-performance VLSI design.

Our approach to speedup the wiring evaluation for such a huge netlist in floorplanning is net reduction. The idea is to group common nets between two blocks into a single net. Before grouping the nets, it is necessary for multi-terminal nets to be decomposed into two-terminal nets first. We use bounded-degree hypergraph-to-graph transformation to accomplish the decomposition. The main objective of this approach is to meet the constraint that the connection of each block is within its pin-limit. It may look like the general connectivity approaches in [12, 13], which are different from this net reduction in terms of the objectives. In [12, 13], they use general connectivity model to estimate the wirelength and timing evaluation during placement. In our approach, however, in addition to get “connectivity matrix”, we perform global routing based on the net regrouping result.

After net reduction, we can use another approach to further reduce the congestion occurred in floorplanning. This approach is based on Lagrangian relaxation and the objective is to minimize the maximum violation against routing resources in a given floorplan. The update of Lagrangian multipliers will help reducing the local congestion in a systematic way, trying to balance the routing among routing regions. Because of the nature of Lagrangian relaxation, that is, it takes longer runtime to converge to optimal solutions, we perform it at the end of the process. Combining the net reduction and Lagrangian relaxation based global router makes the interconnect-centric floorplanning more efficient and effective.

To effectively evaluate wiring results, we give some general definitions to some subjective terms in this paper. In order to estimate congestion/routability, we divide the floorplan into a number of bins by a grid the same way in [2]. For each bin boundary, we define its capacity as the maximum number of nets that can cross it. If the number of nets crossing a bin boundary exceeds the capacity of the bin boundary, there is overflow in that bin boundary. We estimate the maximum overflow among all bin boundaries and total overflow for all overflow occurred as important part of our objectives to evaluate the quality of a floorplan. In the following sections, we describe the details of efficient wiring evaluation in Section 3 and those of effective wiring evaluation in Section 4.

3. FASTER WIRING EVALUATION BASED ON NET REDUCTION

In this section, we present an approach, net reduction, to significantly reduce the runtime of interconnect-centric floorplanning algorithms. Net reduction is a technique to help interconnect cost evaluation for huge netlists. For two-terminal nets, net reduction is easy to be accomplished by grouping them into wider nets (See Figure 1). For multi-terminal nets, however, net decomposition is needed to perform the net reduction. Arbitrary net decomposition will need to pay the price of crowding the connections into a block. As an example, Figure 2 shows a netlist with one two-terminal net and three multi-terminal nets in a floorplan containing six blocks. Without loss of generality, assume that the pin-limit for block b4 is 3. In Figure 3, with the same number of wider nets, (a) shows an illegal net decomposition since the connection of block b4 is 6, which exceeds the pin-limit, while (b) shows a feasible net decomposition.

3.1 Problem Formulation

Based on the objective illustrated above, we can state the net reduction problem as follows. Given a floorplan of n
blocks $b_1, \ldots, b_n$, and their pin-limit $d_1, \ldots, d_n$, respectively, and given a netlist of $m$ nets (multi-terminal and two-terminal) $e_1, \ldots, e_m$, we want to find a net reduction solution such that the resultant nets are two-terminal nets (regrouping to wider nets) and that the solution meets the constraint that the connection for each block is within its pin-limit.

3.2 Algorithm for Net Reduction

Our algorithm for net reduction is similar to the work in [6] for board-level routing for FPGA-based logic emulation. According to the objective in the problem formulation, we observe that if we can model the netlist as a hypergraph by representing the functional blocks in a floorplan as vertices and the nets as hyperedges, the problem of net decomposition can be seen as a bounded-degree hypergraph-to-graph transformation, where the bounded degree is the pin-limit for a block.

We introduce the bounded-degree hypergraph-to-graph transformation, which obtains the degree specifications of nets with respect to each block by optimally solving the problem. We use this transformation to decompose those nets and create the corresponding spanning trees. Next, we contract the net reduction algorithm based on the transformation and resultant spanning trees. The last step of the algorithm is to group all decomposed two-terminal nets into wider interconnect.

3.2.1 Bounded-Degree Hypergraph-to-Graph Transformation

We want to solve the problem of transforming a hypergraph to a graph by modelling each hyperedge as a spanning tree so that the degree of each vertex $v$ in the resultant graph does not exceed some given bound $\sigma_v$. This problem is studied as the bounded-degree hypergraph-to-graph transformation problem. Figure 4 shows a transformation of a hypergraph to a graph where the degrees of all vertices are bounded by 3. Each hyperedge is transformed to a spanning tree that connects all the vertices in the hyperedge. In general, the degree bound $\sigma_v$ can be different for different vertex $v$.

To model a hyperedge of $p (\geq 2)$ vertices as a spanning tree that connects the $p$ vertices, clearly the sum of the degrees of the vertices in the spanning tree must be $2(p-1)$ and the degree of each vertex must be at least one. On the other hand, it can be shown that given any vector $d = (d_1, \ldots, d_p) \in \mathbb{N}^p$ such that $\sum_{i=1}^p d_i = 2(p-1)$ and $d_1, \ldots, d_p \geq 1$, we can always construct a spanning tree of $p$ vertices whose degrees are equal to the $p$ elements of vector $d$. We can easily construct an efficient algorithm for generating a spanning tree given any valid degree specification vector. Furthermore, we can guarantee to get a minimum-height spanning tree, which is good for performance.

Now we describe the algorithm for the bounded-degree hypergraph-to-graph transformation problem. Suppose we are to transform a hypergraph $H = (V, E)$ to a graph $G$ given the degree bound $\sigma_v$ of each vertex $v$ in $V$. We construct a flow network $W = (N, A)$ as follows. The node set $N$ is $\{e_1, e_2, \ldots, e_\mid E\}, v_1, v_2, \ldots, v_\mid V\}$, where node $e_i$ corresponds to hyperedge $e_i$ in $E$ ($i = 1, \ldots, \mid E\}$), node $v_j$ corresponds to vertex $v_j$ in $V$ ($j = 1, \ldots, \mid V\}$), node $s$ is the source, and node $t$ is the sink. For every hyperedge $e_i$, if it connects $p$ vertices, then there is an arc from node $s$ to node $e_i$ with capacity $c(s, e_i) = p - 2$, and for every vertex $v_j$ connected by hyperedge $e_i$, there is an arc from node $e_i$ to node $v_j$ with capacity $c(e_i, v_j) = p - 2$. For every vertex $v_j$ in $V$, there is an arc from node $v_j$ to node $t$ with capac-
\( c(v_j, t) = \sigma_{v_j} - \text{deg}_H(v_j) \), where \( \text{deg}_H(v_j) \) is the degree of vertex \( v_j \) in \( H \). For example, to transform the hypergraph in Figure 4(a) to a graph where the degree of each vertex is bounded by 3, we construct the network shown in Figure 5(a).

To model each hyperedge as a spanning tree so that the total degree of each vertex \( v \) in the resultant graph is bounded by \( \sigma_v \), we have to find an integral maximum flow from \( s \) to \( t \) in the constructed network. It is well-known that if the capacities of all arcs in a network are integers, then there exists an integral maximum flow (i.e., the flow in each arc is an integer). Furthermore, in this case, maximum flow algorithms such as the Ford-Fulkerson method [4] always produce an integral maximum flow. In the following theorem, we show how a feasible transformation can be derived from an integral maximum flow solution.

**Theorem 3.1.** A bounded-degree hypergraph-to-graph transformation problem is feasible if and only if in a maximum flow of the constructed network \( W \), the flow in arcs \((s, e_1), (s, e_2), \ldots, (s, e_{|E|})\) are all at their capacities.  

### 3.2.2 Net Reduction by Multi-terminal Net Decomposition

We use the transformation presented in Section 3.2.1 to construct our net reduction algorithm. An illustrated example is followed by the algorithm.

**Net Reduction Algorithm**

1. Model the netlist as a hypergraph \( H \) by representing the functional blocks as vertices and the nets as hyperedges.
2. Transform hypergraph \( H \) to a graph where the degree of each vertex does not exceed the pin-limit per block:
   (a) Construct network \( W \).
   (b) Find a maximum flow in \( W \).
   (c) Get a valid degree specification vector for each hyperedge from the maximum flow solution.
   (d) Transform each hyperedge into a spanning tree according to its degree specification vector.
3. Decompose each multi-terminal net into subnets according to the corresponding hyperedge to spanning tree transformation.
4. Group all subnets according to the spanning trees to generate a set of wider interconnects.

For example, we are to decompose the multi-terminal nets in a floorplan shown in Figure 2. We first model it as the hypergraph (like Figure 4(a)) where hyperedge \( e_i \) represents net \( i \) and vertex \( v_j \) represents block \( j \). Then we construct the network in Figure 5(a). A maximum flow of the network is found in Figure 5(b). Using flow \( f \), the degree specification vector for hyperedge \( e_2 \) is \( f(e_2, v_1) + 1 \) if one maximum flow saturates arcs \((s, e_1), \ldots, (s, e_{|E|})\), then any other maximum flow also saturates arcs \((s, e_1), \ldots, (s, e_{|E|})\).

1. By the construction of \( W \), if one maximum flow saturates arcs \((s, e_1), \ldots, (s, e_{|E|})\), then any other maximum flow also saturates arcs \((s, e_1), \ldots, (s, e_{|E|})\).

**Figure 5:** (a) Flow network built upon the hypergraph model. (b) An integral maximum flow. The number besides each arc is the (a) capacity/(b) flow in the arc.

**Figure 6:** Spanning trees for hyperedges \( e_2 \) and \( e_3 \).

**4. More Accurate Wiring Evaluation Based on Lagrangian Relaxation**
In this section, a new approach for more accurate wiring evaluation based on *Lagrangian relaxation* is presented. We use the Lagrangian relaxation technique to construct a systematic global router. Based on this technique, updating the Lagrangian multipliers is the guidance to help the router to alleviate local congestion in routing regions.

### 4.1 Problem Formulation

We are given \( m \) nets and \( n \) bin boundaries in routing grid. The decision vector \( \mathbf{x} \) is defined as follows. Let \( x_{ik} \) be a decision variable of each bin boundary \( i \) such that

\[
x_{ik} = \begin{cases} 
1 & \text{if net } k \text{ cross bin boundary } i \\
0 & \text{otherwise}
\end{cases}
\]

Then the global routing problem can be formulated as the following *primal problem*:

\[
\text{Minimize } O_{\text{max}} \\
\text{Subject to } \sum_k x_{ik} \leq C_i + O_{\text{max}}, \quad i = 1, 2, \ldots, n
\]

where \( O_{\text{max}} \) is the maximum overflow/violation among all bin boundaries, and \( C_i \) is the capacity for the \( i \)th bin boundary. The objective is to minimize the maximum overflow in routing grid so that local congestion is minimized.

### 4.2 Lagrangian Relaxation

Lagrangian relaxation is a general technique for solving optimization problems with difficult constraints [1]. According to the Lagrangian relaxation procedure, we introduce non-negative multipliers, called *Lagrangian multipliers*, to the constraints in order to get rid of those difficult constraints and incorporate them into the objective function. Let \( \lambda_i \) denote the multiplier for the constraint \( \sum_k x_{ik} \leq C_i + O_{\text{max}} \), \( \mathbf{\lambda} \) be the vector of all Lagrangian multipliers introduced to the constraints. Then the Lagrangian relaxation subproblem (LRS) associated with the multiplier \( \mathbf{\lambda} \) becomes:

\[
\text{Minimize } O_{\text{max}} + \sum_i \lambda_i (\sum_k x_{ik} - C_i - O_{\text{max}})
\]

Let \( L(\mathbf{\lambda}) \) be the amount above that we are trying to minimized. For any value of the Lagrangian multiplier \( \mathbf{\lambda} \), \( L(\mathbf{\lambda}) \) is a lower bound of the optimal objective function value of the original problem. To obtain the sharper possible lower bound, we need to solve the following problem

\[
\text{Maximize } L(\mathbf{\lambda}) \\
\text{Subject to } \mathbf{\lambda} \geq 0
\]

which we refer to as the *Lagrangian multiplier problem* or *Lagrangian dual problem* (LDP) of primal problem. In the following sub-sections, we present the approaches to solve the primal problem by solving LRS and LDP.

#### 4.2.1 Simplification of Lagrangian Relaxation Subproblem

The Lagrangian relaxation subproblem (LRS) can be simplified by applying Kuhn-Tucker conditions.

\[
L(\mathbf{\lambda}) = O_{\text{max}} + \sum_i \lambda_i (\sum_k x_{ik} - C_i - O_{\text{max}})
\]

\[
= \sum_i \lambda_i (\sum_k x_{ik} - C_i) + O_{\text{max}} (1 - \sum_i \lambda_i)
\]

The Kuhn-Tucker conditions imply that \( \partial L(\mathbf{\lambda}) / \partial O_{\text{max}} = 0 \) for all \( 1 \leq i \leq n \) at the optimal solution of the primal problem. Therefore, in searching for \( \mathbf{\lambda} \) to optimize LDP, we only need to consider the situation such that the conditions are satisfied. Thus, we get the following equation by applying Kuhn-Tucker condition:

\[
\sum_i \lambda_i = 1
\]

That is,

\[
\sum_i \lambda_i = 1
\]

We use \( \Phi \) to denote the \( \mathbf{\lambda} \) satisfying the above relationship for a given routing solution. If \( \mathbf{\lambda} \in \Phi \), the objective function \( L^*(\mathbf{\lambda}) \) becomes:

\[
L^*(\mathbf{\lambda}) = \sum_i \lambda_i \sum_k x_{ik} - \sum_i \lambda_i C_i
\]

where \( \sum_i \lambda_i C_i \) is a constant for given \( \mathbf{\lambda} \) and capacities in all bin boundaries.

#### 4.2.2 Solving Lagrangian Relaxation Subproblem

As mentioned above, we only need to consider solving the Lagrangian relaxation subproblem when \( \mathbf{\lambda} \in \Phi \). We use a reasonable heuristic to find the solution for LRS. From the equation in previous sub-section, the cost of the objective function depends on the term \( \sum_i \lambda_i \sum_k x_{ik} \). A simple idea to route the nets would be trying to minimize this amount: for each two-terminal net, pick the minimum cost path to route, where the cost \( \varphi = \sum_i \lambda_i \sum_k x_{ik} \). As in [2], we use L-shaped and Z-shaped routing instead of general maze routing.

#### 4.2.3 Solving Lagrangian Dual Problem

In order to maximize \( L(\mathbf{\lambda}) \) in Lagrangian dual problem (LDP), we only need to consider those \( \mathbf{\lambda} \in \Phi \). First, the approach is to use the solutions of the Lagrangian relaxation subproblem to update the multipliers until the result converges. Here we use subgradient optimization technique to search for “optimal” \( \mathbf{\lambda} \).

\[
\mathbf{\lambda} = [\lambda_k + \theta_k (\sum_k (x_{ik} - C_i - O_{\text{max}}))]^+
\]

where

\[
[x]^+ = \begin{cases} 
x & \text{if } x > 0 \\
0 & \text{if } x \leq 0
\end{cases}
\]

and \( \theta_k \) is a step length at the \( k \)th iteration such that

\[
\lim_{k \to \infty} \theta_k = 0 \text{ and } \sum_{k=1}^{\infty} \theta_k = \infty.
\]
Second, in order to make sure the new updated multiplier \( \tilde{X} \in \Phi \), we need to project the updated multiplier \( \tilde{X} \) back to nearest point \( X^* \) in \( \Phi \).

5. EXPERIMENTAL RESULTS AND CONCLUDING REMARKS

We have tested our approaches on 33 blocks with 5K, 10K, and 15K nets, which are generated artificially from am33 in MCNC benchmarks to test on huge netlists. All the experiments were carried out on a 300MHz Pentium II Intel Processor. Table 1 shows the runtime speedup by using the new improved approach in wiring evaluation. We use this approach on the work in [2] with all test sets in the following way. As in [2], we perform accurate interconnect planning as follows: applying bounding-box wirelength estimation during high temperature, apply pin assignment and simple geometry routing during medium and low temperature. In order to justify the effectiveness of our approaches, we perform the more accurate global router on the original netlist in both results, which are from [2] and ours. The new improved approach spent less than 50 minutes to finish while [2] took more than 23 hours to finish in a 15K-net test set, for example. The qualities of the floorplans obtained are comparable in area and routability. On the other hand, we get improvement by using more accurate global router in congestion issue for 41.9% average among 5 MCNC benchmarks in terms of maximum overflow. Note that the network flow based algorithm takes some CPU time, which is a much smaller part compared with longer duration in interconnect-centric floorplanning.

<table>
<thead>
<tr>
<th>Approach</th>
<th>5K nets Time (sec)</th>
<th>10K nets Time (sec)</th>
<th>15K nets Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floorplanner in [2]</td>
<td>19874</td>
<td>60124</td>
<td>85026</td>
</tr>
<tr>
<td>New Approach</td>
<td>1864</td>
<td>2478</td>
<td>2938</td>
</tr>
</tbody>
</table>

Table 1: Speedup comparisons between floorplanner in [2] and our approaches using net reduction and Lagrangian relaxation global router.

In conclusion, we present a simple yet effective idea to significantly reduce the runtime of interconnect-centric floorplanning algorithms. Network flow based net decomposition algorithm is used to decompose multi-terminal nets without violating the pin-limit constraint in each block. We also use the Lagrangian relaxation paradigm to obtain an effective global router in minimizing the maximum overflow in routing region. Combining these two approaches makes the interconnect-centric floorplanning more efficient and effective.

6. REFERENCES


