OPTIMAL ROUTING ALGORITHMS FOR PIN CLUSTERS IN HIGH-DENSITY MULTICHIP MODULES [*]

Muhammet Mustafa Ozdal [†]
Intel Corporation
Hillsboro, OR 97124
mustafa.ozdal@intel.com

Martin D. F. Wong
Univ. of Illinois at U-C
Urbana, IL 61801
mdfwong@uiuc.edu

Philip S. Honsinger
IBM Microelectronics
Hopewell Junction, NY 12533
honsinge@us.ibm.com

ABSTRACT

As the circuit densities and transistor counts are increasing, the package routing problem is becoming more and more challenging. In this paper, we study an important routing problem encountered in typical high-end MCM designs: routing within dense pin clusters. Pin clusters are often formed by pins that belong to the same functional unit or the same data bus, and can become bottlenecks in terms of overall routability. Typically, these clusters have irregular shapes, which can be approximated with rectilinear convex boundaries. Since such boundaries have often irregular shapes, a traditional escape routing algorithm may give unroutable solutions. In this paper, we study how the positions of escape terminals on a convex boundary affect the overall routability. For this purpose, we propose a set of necessary and sufficient conditions to model routability outside a rectilinear convex boundary. Given an escape routing solution, we propose an optimal algorithm to select the maximal subset of nets that are routable outside the boundary. After that, we focus on an integrated approach to consider routability constraints (outside the boundary) during the actual escape routing algorithm. Here, we propose an optimal algorithm to find the best escape routing solution that satisfies all routability constraints. Our experiments demonstrate that we can reduce the number of layers by 17% on the average, by using this integrated methodology.

1. INTRODUCTION

As the circuit densities and transistor counts are increasing, the package routing problem is becoming more and more challenging. In the current industrial designs, the limitations of commonly used rip-up and reroute methodologies [1, 5, 9] are becoming more significant [8]. So, new routing algorithms are needed to handle new challenges effectively. One of the most difficult parts of the package routing problem is routing within dense pin clusters [7]. Both packaging hierarchy and functional hierarchy imply potential pin clustering at hierarchical interfaces. These clusters are formed typically by pins that belong to the same functional unit or the same data bus. The highest wire demand is typically within such pin clusters and in proximity of the cluster perimeters. As additional objectives (such as delay and noise optimizations) impose further constraints on the routing problem, getting the connections started correctly from the clustered pin areas is becoming an increasingly important issue.

[*]This work was partially supported by the National Science Foundation under grant CCR-0306244 and by an IBM Faculty Award.
[†]This work was done while the author was a graduate student at the University of Illinois at Urbana-Champaign.

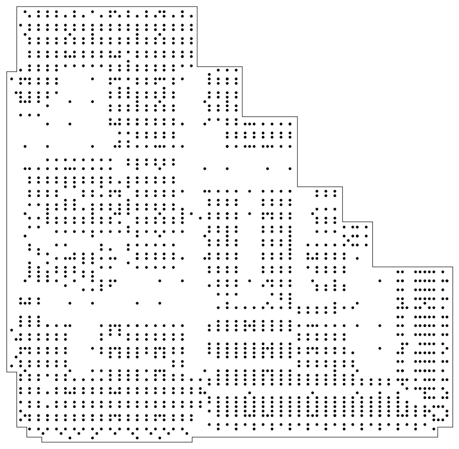
Figure 1: A cluster of pins from a chip mounted on a ceramic MCM module from a real IBM design. The convex boundary enclosing the pin cluster is also illustrated.

A cluster of pins from a real MCM design (from IBM) is illustrated in Figure 1. As seen in this figure, these pin clusters typically have irregular shapes. The empty areas in these clusters can be due to islands of voltage pins, which are routed in dedicated voltage layers. They can also be due to blind or buried vias that do not span all layers of the package. In surface mount type (SMT) components, such as ball grid arrays (BGAs), I/O signals are typically transferred to inner component layers using blind or buried vias [2, 11]. These vias are used to carry the I/O signals from bare chips (on the top layer) to the layers on which the corresponding nets are routed. In other words, once a net is routed on one layer, its pin is not extended further down the layer stack. As a result, the cluster of pins typically *shrinks* as we go further down the layer stack. Due to all these factors, the pin clusters often have irregular shapes, as shown in Figure 1.

The escape routing problem has been studied extensively in the literature [3, 4, 6, 10, 12] to route nets from individual pins to a boundary. However, a rectangular boundary is assumed in these algorithms most of the time, and the effects of irregular boundaries are not considered. Normally, a traditional escape routing algorithm can also be applied on a pin cluster with an irregular

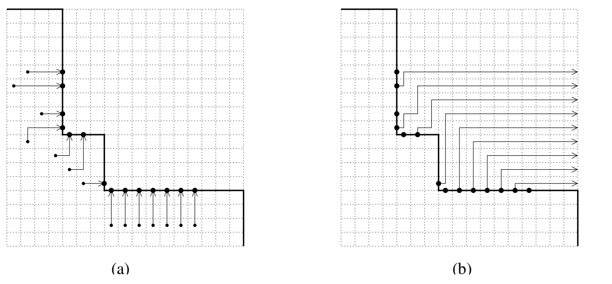

Figure 2: (a) An escape routing solution for 14 nets from pins to a convex boundary. (b) Only 9 out of 14 escaped nets can be routed outside due to conflicts with each other.

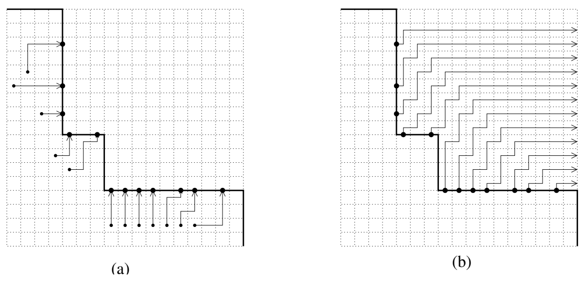

Figure 3: (a) A different escape routing solution for the problem of Figure 2. (b) All 12 nets are routable outside the boundary.

boundary. However, the problem here is that routability outside the boundary is not guaranteed, since the routes of nets that escape to this boundary may conflict with each other outside. Figure 2 gives a small example to illustrate this problem in more detail. Assume that a set of nets have been routed to a set of escape terminals on the boundary, as shown in Figure 2(a). Here, one problem is how to determine whether all nets that have escaped to the boundary can be successfully routed outside, since some of the escaped nets can conflict with each other. In this example, there are 14 nets that have successfully escaped to the boundary; however only 9 of them can be successfully routed outside, as shown in Figure 2(b). In Section 3, we propose a set of necessary and sufficient conditions to determine routability based on only the positions of escape terminals on an arbitrary convex boundary. Another problem here is how to determine the maximal routable subset if a given set of escape terminals is not routable. A maximal routable subset is shown in Figure 2(b), together with a feasible routing solution, corresponding to the escape terminals in Figure 2(a). For this purpose, we propose an optimal algorithm in Section 4. In this algorithm, the optimal subset is determined based on only the positions of the escape terminals, without performing any routing outside the boundary. After that, we focus on an integrated approach in Section 5 to consider *routability outside* during the actual escape routing algorithm. In other words, instead of using a two-step methodology (escape routing followed by routability analysis), we directly find the escape routing solution such that routability outside is also guaranteed. For example, Figure 3(a) shows a different escape routing solution for the problem in Figure 2. Here, all the nets that have escaped are routable, as shown in Figure 3(b). The proposed algorithm for this purpose is also proven to be optimal.

The rest of this paper is organized as follows. We give a formal description of this problem in Section 2. Then in Section 3, we propose a set of necessary and sufficient conditions that exactly model routability outside the given convex boundary. Based on these constraints, we propose an optimal algorithm in Section 4 to select the maximal subset of routable escape terminals. After that, we propose an integrated approach in Section 5 that incorporates the routability constraints into the original escape routing algorithm in an optimal way. In section 6, we present our experimental results, and demonstrate the effectiveness of our algorithms.

2. PROBLEM FORMULATION

Let $\mathcal{P}$ denote a cluster of pins, and let B denote the rectilinear convex boundary enclosing $\mathcal{P}$. Our purpose is to find a routing solution from each pin in $\mathcal{P}$ to an *escape terminal* on B. Here, the scarcity of routing resources inside dense pin clusters do not allow usage of additional buried vias. Hence, the escape routing solution needs to be planar on every layer.

As illustrated in Figure 2, the nets escaping to an irregular boundary can have conflicts with each other outside. A given escape routing solution is defined to be *routable outside* iff all nets escaping to boundary B can be routed without conflicts, as illustrated in Figure 3(b). Here, let us assign a unique index to every escape terminal on boundary B, as shown in Figure 4. Furthermore, let $\#t(x, y)$ denote the number of nets escaping to escape terminals in the interval $[x, y]$, e.g. $\#t(8, 13) = 5$ in part (a), and $\#t(8, 13) = 3$ in part (b) of Figure 4. Here, the first problem we focus on is how to determine whether a given escape routing solution is also *routable outside*, using these $\#t(x, y)$ values. If the given solution is not routable outside, the next problem becomes how to select the maximal subset of routable escape terminals. Finally, the third problem is how to find an escape routing solution in an optimal way such that overall routability is guaranteed. We study these problems in this paper, and propose models and algorithms to solve each of them optimally.

For simplicity of presentation, we will consider only a single layer. In other words, our objective will be to find the maximal escape routing solution on one layer, given a set of candidate pins. It is possible to process layer by layer, and apply our algorithms on every layer. However, our algorithms can also be extended to multilayer problems in a straightforward way, by duplicating the given constraints for every layer.

3. CONSTRAINT MODELING

Our purpose in this section is to investigate the relationship between escape terminal positions on a given convex boundary and the overall routability. For this, we define a set of necessary and sufficient conditions that exactly model *routability outside* the boundary. For simplicity of presentation, we will first focus on a single corner of a given convex boundary in Section 3.1, and then generalize this model for an arbitrary convex boundary in Section 3.2. This constraint modeling will be especially useful since it can be incorporated into the original escape routing algorithm in such a way to guarantee routability outside the boundary.

3.1. Corner Constraints

In this section, we will consider a boundary with a single corner, as shown in Figure 4. Here, let r and $r + 1$ denote the escape terminals on the corner, and let k denote the width of one side of the

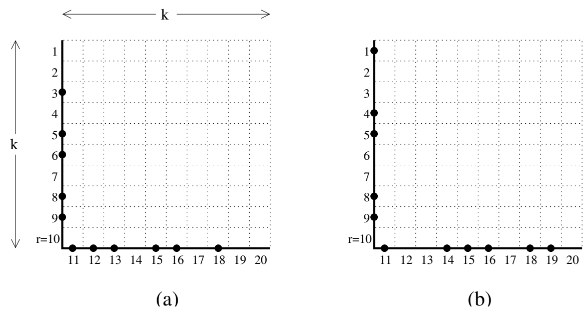

Figure 4: A boundary with a single corner is illustrated, where filled circles represent the escape terminals at which an escape route ends (the escape routes inside are not shown for clarity). Two examples with different terminals are given in parts (a) and (b).

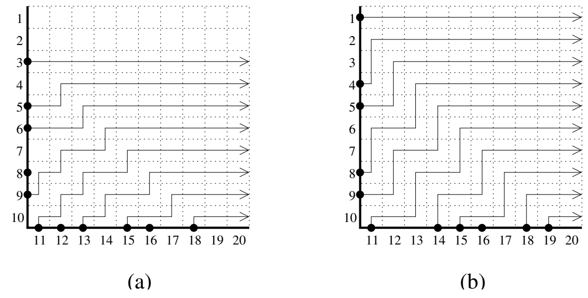

Figure 5: Routing solutions outside the boundaries for the problem given in Figure 4. Three and one nets are unroutable in the solutions of parts (a) and (b), respectively.

corner[1]. Furthermore, let $\#t(x, y)$ denote the number of nets that have escaped to terminals in the interval $[x, y]$. Observe in the example of Figure 4 that $r = 10$, and $k = 10$. Also, $\#t(9, 12) = 3$ in part (a), and $\#t(9, 12) = 2$ in part (b), etc. The following theorem defines the necessary and sufficient conditions for routability:

Theorem 3.1. *An escape routing solution is routable if and only if $\#t(r\text{-}i{+}1, r{+}i) \le i$, for each i, $1 \le i \le k$. In other words, routability is guaranteed if and only if the number of nets escaping to terminals in the interval $[r\text{-}i{+}1, r{+}i]$ is less than or equal to i, for each i.*

As an example, let us consider the boundary given in Figure 4, where the escape terminals are marked from 1 to 20. Here, the following conditions are necessary and sufficient for routability: $\#t(10, 11) \le 1$, $\#t(9, 12) \le 2$, ..., $\#t(1, 20) \le 10$. The given escape routing solution in part (a) violates the conditions $\#t(9, 12) \le 2$, $\#t(8, 13) \le 3$, and $\#t(5, 16) \le 6$; hence 3 out of 11 nets are unroutable, as illustrated in Figure 5(a). Similarly, the solution in part (b) violates the condition $\#t(5, 16) \le 6$, resulting in one unroutable net.

PROOF. NECESSITY: We first prove that the constraints given in Theorem 3.1 are necessary for routability outside. For any i value, let D_i denote the diagonal line spanning the grid cells that have Manhattan distance of i to the corner, as shown in Figure 6(a). It is obvious that the number of *outlets* (i.e., grid cells through which nets can escape) on D_i is equal to i. Since a net escaping to a terminal in the interval $[r - i + 1, r + i]$ must use an outlet on D_i, the necessity of constraint $\#t(r - i + 1, r + i) \le i$ follows. □

[1]For simplicity, assume that the widths of both sides are equal as shown in Figure 4. The generalization will be given in Section 3.2.

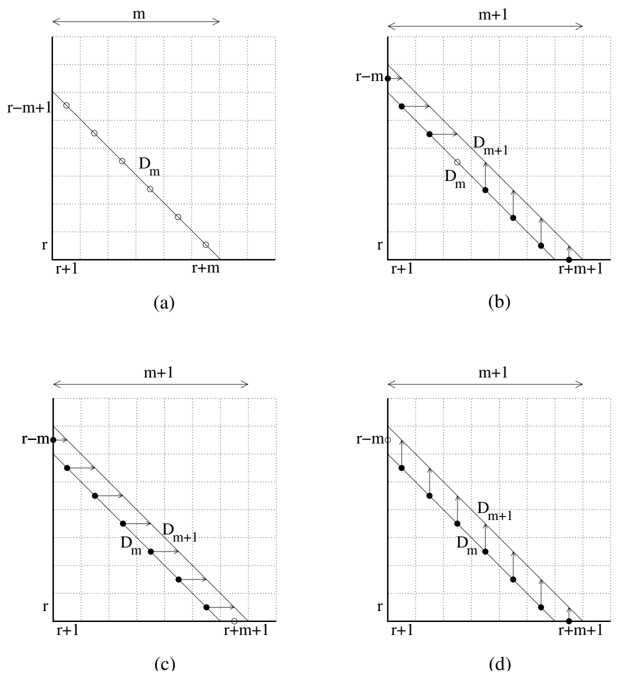

Figure 6: (a) Diagonal D_m has m escape outlets (shown as hollow circles). (b) If there is an unused outlet on D_m, all nets are routable to D_{m+1}, even if both escape terminals $r - m$ and $r + m + 1$ are selected. (c,d) If all outlets on D_m are occupied, then routability is guaranteed as long as at most one of terminals $r - m$ and $r + m + 1$ is selected.

PROOF. SUFFICIENCY: Let us make the following inductive hypothesis: *If the constraints $\#t(r - i + 1, r + i) \le i$ are satisfied for each i, $1 \le i \le k$, then all nets escaping to terminals in the interval $[r - k + 1, r + k]$ can escape to diagonal D_k.* Again, D_k denotes the diagonal line spanning the grid cells that have Manhattan distance of k to the corner, as shown in Figure 6(a). It is straightforward to show that this hypothesis holds for the base case $k = 1$. Now let us assume that it holds for $k = m$, and we will prove it for $k = m + 1$. For this, we need to consider two cases:

- Case 1: $\#t(r - m + 1, r + m) < m$. Here, since there are less than m outlets used on diagonal D_m, there is at least one outlet unused (shown as a hollow circle in Figure 6(b)). Even if there are two nets escaping to terminals $r - m$ and $r + m + 1$, all nets will still be routable to diagonal D_{m+1}, as shown in Figure 6(b).
- Case 2: $\#t(r - m + 1, r + m) = m$. For the constraint $\#t(r - m, r + m + 1) \le m + 1$ be satisfied, only one net can escape to terminals $r - m$ and $r + m + 1$. As shown in Figure 6(c), and (d), all nets will still be routable

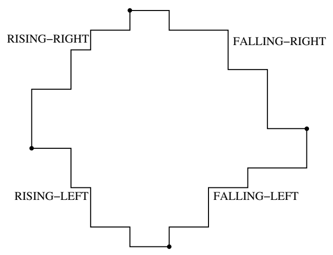

Figure 7: Different boundary regions of a rectilinear convex boundary are illustrated.

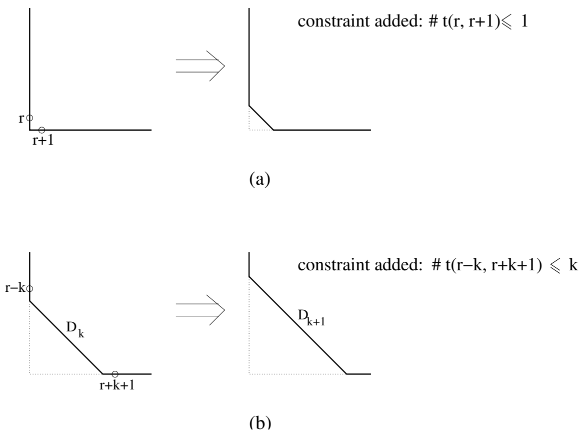
constraint added: # t(r, r+1) ⩽ 1
r
r+1
(a)
constraint added: # t(r−k, r+k+1) ⩽ k
r−k
D k
r+k+1
D k+1
(b)

Figure 8: Illustration of the boundary transformations given in (a) Definition 3.2, and (b) Definition 3.3. The corresponding constraints generated are also shown.

to diagonal D_{m+1}, as long as only one of the terminals $r-m$ or $r+m+1$ is selected.

So, the inductive proof is complete. □

3.2. Generalization to Arbitrary Convex Boundaries

In this section, the idea presented in Section 3.1 is generalized to arbitrary convex boundaries. For a rectilinear convex boundary, we can make the following definition:

Definition 3.1. *A rectilinear convex boundary is defined to have four different regions: falling-right, falling-left, rising-left, and rising-right regions, as illustrated in Figure 7.*

It is obvious that nets escaping to one boundary region (e.g. falling-right region) do not interfere with nets escaping to other regions outside the boundary. In other words, we can consider each of falling-right, falling-left, rising-left, and rising-right regions independent of each other while determining routability outside the boundary. So, in the rest of this section, we will propose the necessary and sufficient conditions for only a falling-right boundary region. It is straightforward to generalize these conditions for other region types.

For the ease of presentation, we will define the routability conditions using the algorithm given in Figure 10. This algorithm is based on boundary transformations that are defined in Definitions 3.2 and 3.3. It is important here to note that these are only

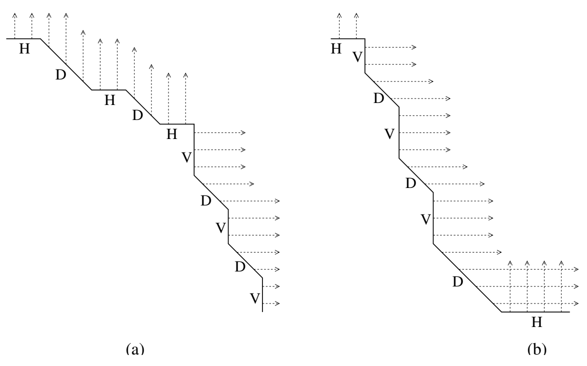

Figure 9: Illustration of routing conflicts outside a falling-right boundary. (a) There is no H-segment after a V-segment; hence conflict-free routing is possible outside. (b) The H-segment after V-segment causes routing conflicts.

conceptual transformations used for the purpose of presentation. In other words, these transformations are not actually performed (i.e., the original boundary still remains intact); however they are used conceptually to generate the set of necessary and sufficient conditions for routability. In the following, let H-segment, V-segment, and D-segment denote horizontal, vertical, and diagonal boundary segments, respectively.

Definition 3.2. *Consider a corner of a falling-right boundary where a V-segment is followed by an H-segment. We can (conceptually) transform this corner as shown in Figure 8(a), and add the explicit constraint $\#t(r, r+1) \leq 1$, where the escape terminals on the corner are denoted as r and $r+1$. Intuitively, replacing a corner with a diagonal as in this figure implies that we don't have to consider this corner anymore for routability analysis, as long as the constraint $\#t(r, r+1) \leq 1$ is satisfied.*

Definition 3.3. *Consider a falling-right boundary that contains a V-segment, followed by a D-segment, followed by an H-segment. We can (conceptually) transform this boundary as shown in Figure 8(b), and add the explicit constraint $\#t(r-k, r+k+1) \leq k+1$, where r and k are as defined in this figure.*

Lemma 3.2. *Let B denote the original escape boundary, and let B' denote the boundary after one of the transformations given in Definitions 3.2 and 3.3 is applied on B. The routability characteristics of B is equivalent to the routability characteristics of B' iff the additional constraint introduced during the transformation is satisfied.*

PROOF. The proof is very similar to the inductive proof of Theorem 3.1. □

Intuitively, we can continue performing boundary transformations, and defining new conditions, until the transformed boundary is guaranteed to be routable. The following lemma states the routability condition for a falling-right boundary.

Lemma 3.3. *A falling-right boundary B is guaranteed to be routable outside if there is no H-segment after a V-segment in B.*

PROOF. Figure 9 shows the main intuition. In part (a), there is no H-segment after a V-segment, and all nets escaping to all

```
CREATE-CONSTRAINT-FOREST(falling-right boundary $B$)
  $\mathcal{C} \leftarrow \emptyset$        // the set of necessary and sufficient conditions
  while there is no H-segment after a V-segment in $B$
      perform a (conceptual) boundary transformation
      add the corresponding constraint into $\mathcal{C}$.
  create the constraint forest $\mathcal{F}$ as follows:
      for each constraint in $\mathcal{C}$, a node exists in $\mathcal{F}$.
      Node $u$ is a parent of node $v$ in $\mathcal{F}$ iff $u$ has the smallest
            interval that is a proper superset of $v$'s interval.
  return $\mathcal{F}$
```

Figure 10: Algorithm to generate the set of necessary and sufficient conditions for a given falling-right boundary.

terminals on the boundary are routable. On the other hand, there is an H-segment after a V-segment in part (b), and routing conflicts are possible outside the boundary. □

Figure 10 gives the algorithm we use to generate the set of necessary and sufficient conditions corresponding to a given falling-right boundary. This set of conditions is represented as a *constraint-forest* $\mathcal{F}$, where each node in $\mathcal{F}$ corresponds to a constraint in the form $\#t(x,y) \leq z$, i.e., the number of nets escaping to terminals in the interval $[x,y]$ is less than or equal to z. Here, if node u is a parent of node v, then the constraint interval corresponding to node u is guaranteed to be a proper superset of the constraint interval corresponding to node v. Figure 11 illustrates the constraint forest generation process with an example.

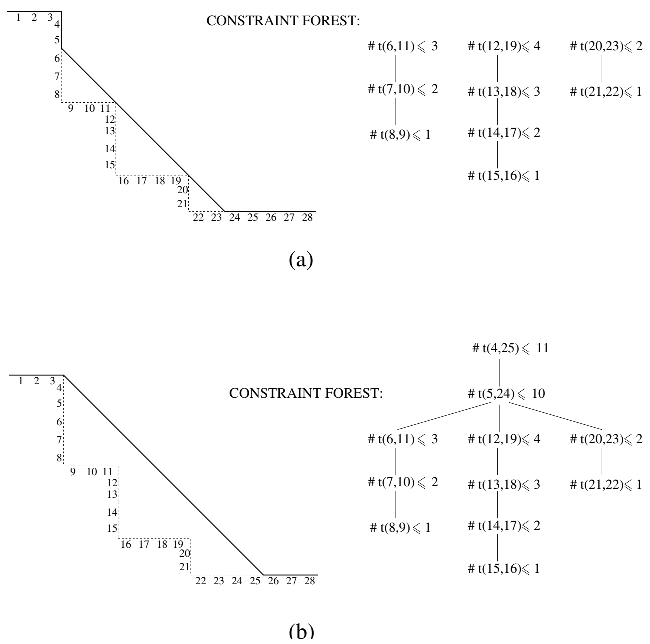

Figure 11: Illustration of constraint forest generation on a convex boundary with 28 escape terminals. The original boundary is shown with dotted lines. (a) The boundary after the first set of transformations, and the corresponding partial forest. (b) The final boundary, and the constraint forest generated.

4. SELECTION OF MAXIMAL ROUTABLE ESCAPE TERMINALS

In this section, we assume that escape routing to an arbitrary convex boundary has already been performed, and our purpose is to select the maximum subset of terminals that can be routed outside without any conflicts. For this, we make use of constraint forest $\mathcal{F}$, which was defined in Section 3. Before giving the details of this algorithm, we will make some observations about the properties of $\mathcal{F}$ as follows.

Remark 4.1. *Consider two nodes u and v in constraint forest $\mathcal{F}$. If u is an ancestor of v, then the interval corresponding to u is a proper superset of the interval corresponding to v.*

Remark 4.2. *Consider two nodes u and v in constraint forest $\mathcal{F}$. If u is neither ancestor nor descendant of v, then the intervals corresponding to u and v do not overlap.*

Remark 4.3. *Consider a non-leaf node u that has the constraint $\#t(x,y) \leq z$, The union of the constraints corresponding to all children of node u is equivalent to $\#t(x+1,y-1) \leq z-1$.*

Remark 4.4. *The number of nodes in constraint forest $\mathcal{F}$ is linear in the number of escape terminals on the boundary.*

These observations directly follow from the definition of the constraint forest. Readers can refer to Figure 11 for an example.

The algorithm we propose for selection of maximal routable escape terminals is given in Figure 12. The recursive function given in this figure needs to be called for each root node in the constraint forest $\mathcal{F}$. Intuitively, we first process the children of the current node r, and find the maximal set of escape terminals that satisfy the descendant constraints. Then, we consider the constraint at r, which is $\#t(x,y) \leq z$. From Remark 4.3, we know that the escape terminals in the interval $[x+1, y-1]$ have already been processed by the descendants of node r. So, we consider only the escape terminals x and y here. If the size of the selected terminal set T is still less than z, then we add these terminals to T, making sure that the constraint $\#t(x,y) \leq z$ is not violated. The following theorem states the optimality and the time complexity of this algorithm.

Theorem 4.5. *The algorithm proposed in Figure 12 returns the maximal subset of escape terminals that are routable outside the boundary. The time complexity of this algorithm is linear in the number of escape terminals on the boundary.*

PROOF. The time complexity of the algorithm directly follows from Remark 4.4, since each node in the forest is processed only once. For optimality, first let us consider Remark 4.2, which indicates that different subtrees rooted at r specify constraints for non-overlapping intervals. In other words, terminal selection in each subtree can be performed independent of each other. Now, we will prove the optimality of this algorithm using induction. As the base case let us consider a forest consisting of only leaf nodes. It is obvious that our algorithm will give the optimal solution, since each leaf is independent of each other (due to Remark 4.2). Now, assume that the inductive hypothesis holds for each child subtree of node r, i.e., each recursive call to a child of r returns the optimal solution. Let us denote the constraint corresponding to node r as $\#t(x,y) \leq z$. We know that the intervals corresponding to different subtrees do not overlap with each other (due to Remark 4.2), and the union of the intervals considered in r's child

```
SELECT-ESCAPE-TERMINALS(Node r)
  $T \leftarrow \emptyset$            // the selected terminal set
  for each child $u$ of $r$ do
      $T \leftarrow T \cup$ SELECT-ESCAPE-TERMINALS(u)
  Let $\#t(x, y) \leq z$ be the constraint corresponding to $r$
  If there is an escape route ending at terminal $x$
      $T \leftarrow T \cup \{x\}$
  If there is an escape route ending at terminal $y$
      if $|T| < z$
          $T \leftarrow T \cup \{y\}$
  return $T$
```

Figure 12: The algorithm to select the maximal routable escape terminals. This algorithm needs to be called for each root node in the constraint forest.

subtrees is $[x+1, y-1]$ (due to Remark 4.3). From the inductive hypothesis, we can state that after the recursive calls, T contains the maximal routable set of escape terminals in the interval $[x+1, y-1]$. So, while processing node r, we only need to consider whether we should add escape terminals x and y into T. Note that the maximum number of escape terminals that can be selected in the interval $[x, y]$ is z due to the constraint at node r. Now, let us consider two cases: (1) $\#t(x+1, y-1) < z-1$, and (2) $\#t(x+1, y-1) = z-1$. In the first case, both x and y can be added to T, if there are escape routes ending at these terminals; hence the optimal solution in the interval $[x, y]$ is obtained. In the second case, we need to make sure that the number of selected terminals does not exceed z before selecting terminals x or y. However, we know that the maximum size of T can be z in any routable solution; hence the optimal solution in the interval $[x, y]$ is still maintained. So, our inductive proof is complete. $\square$

5. ROUTABILITY-DRIVEN ESCAPE ROUTING

In the previous sections, we have assumed that the escape routing solution has already been found, and we have proposed a set of constraints to determine the routability outside the boundary. In this section, we propose an integrated approach to solve the escape routing problem in such a way that routability outside is guaranteed. For this purpose, we define a flow network corresponding to the constraint forest proposed in Section 3, and then we augment it to the original flow network which corresponds to the escape routing problem.

It is well known that the problem of escape routing can be solved optimally using network flow [4]. In the literature, there have also been different improvements proposed for the purpose of reducing execution time and space requirements [3, 6]. Our constraint models can be applied to different flow models; however we will focus on the basic network flow formulation for simplicity of presentation.

Let us assume that flow network $\mathcal{N}$ is modeled corresponding to the original escape problem (to a convex boundary) as follows: For each grid cell, there is a vertex in $\mathcal{N}$, with node capacities equal to 1. The vertices corresponding to the neighboring grid cells are connected by edges in $\mathcal{N}$. Furthermore, there are two special vertices: the *source* and the *sink* vertices in the flow network. There is an edge from the *source* vertex to every vertex that corresponds to a grid cell on which a net terminal exists. Similarly, there is an edge to the *sink* vertex from every vertex that corresponds to a grid cell on the boundary. It is known that the maximum flow from the *source* vertex to the *sink* vertex in $\mathcal{N}$ gives the optimal solution for the escape routing problem. Further details about this basic network flow formulation can be found in [4]. Note here that this formulation does not consider the routability constraints outside the boundary, and it is possible to obtain a routing solution that is not routable outside the convex boundary, as illustrated in Figure 2. For the purpose of incorporating the routability constraints, we define the following flow network $\mathcal{N}_C$:

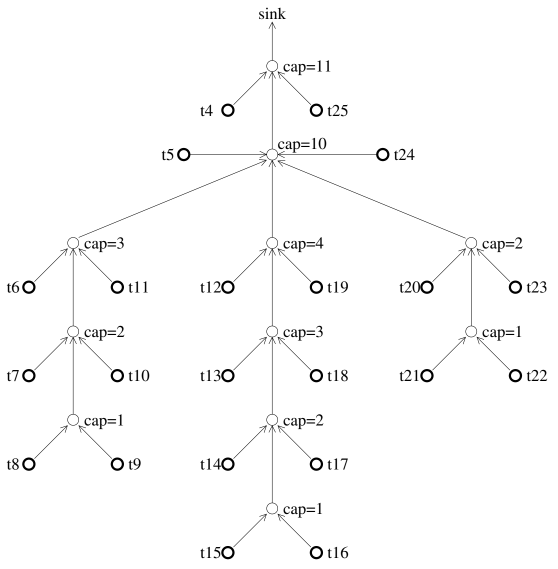

Figure 13: The flow network $\mathcal{N}_C$ corresponding to the constraint forest given in Figure 11(b). The dark and light circles represent t-vertices and c-vertices, respectively. The capacities of c-vertices, and the terminal indices of t-vertices are also shown.

Definition 5.1. *The flow network $\mathcal{N}_C$ corresponding to constraint forest $\mathcal{F}$ is created as follows:*

- *Create a t-vertex corresponding to each escape terminal on the convex boundary. Set the capacity of each t-vertex to 1.*
- *Create a c-vertex corresponding to each node in the constraint forest $\mathcal{F}$.*
- *Consider each c-vertex v_c, which corresponds to the constraint $\#t(x, y) \leq z$. Set the capacity of v_c to z. Then, create the incoming edges to v_c as follows:*
    - *Create an edge to v_c from t-vertex corresponding to escape terminal x.*
    - *Create an edge to v_c from t-vertex corresponding to escape terminal y.*
    - *Create edges to v_c from the c-vertices that correspond to children of v_c (in the constraint forest $\mathcal{F}$).*
- *Consider each c-vertex v_r that corresponds to a root node in the constraint forest $\mathcal{F}$. Create an edge from v_r to sink vertex of $\mathcal{F}$.*

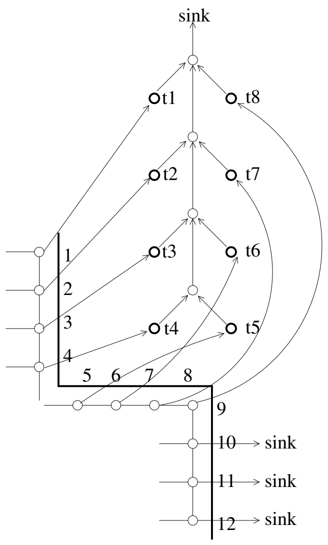

Figure 14: An example illustrating how to augment constraint network $\mathcal{N}_C$ to the original flow network $\mathcal{N}$. Here, an edge exists from each terminal vertex in $\mathcal{N}$ to the corresponding t-vertex in $\mathcal{N}_C$.

The flow network corresponding to the constraint forest of Figure 11(b) is illustrated in Figure 13 as an example. Note that the size of $\mathcal{N}_C$ is linear in the number of escape terminals on the convex boundary, due to Remark 4.4.

Definition 5.2. *The flow network $\mathcal{N}_C$ (corresponding to constraint forest $\mathcal{F}$) can be augmented to the original flow network $\mathcal{N}$ (corresponding to the escape routing problem inside the convex boundary) as follows:*

Consider each vertex v in $\mathcal{N}$ that corresponds to an escape terminal on the boundary. If this escape terminal has a constraint associated with it in constraint forest $\mathcal{F}$, then:

- *The edge from v to the sink vertex is removed.*
- *An edge is created from v to the corresponding t-vertex in $\mathcal{N}_C$.*

This augmentation process is illustrated in Figure 14 with a simple example. Here, the terminals on the corner (terminals 1-8) are connected to the corresponding t-vertices in $\mathcal{N}_C$. Since no constraint is associated with terminals 9-12, they are still connected to the *sink* vertex directly.

Theorem 5.1. *Let $\mathcal{N}$ denote the original flow network corresponding to the escape problem inside the boundary. Let $\mathcal{N}_C$ denote the constraint flow network as given in Definition 5.1. Assume that we augment $\mathcal{N}_C$ to $\mathcal{N}$ as described in Definition 5.2 to obtain the final flow network $\mathcal{N}_F$. The maximum flow on $\mathcal{N}_F$ will give the optimal escape routing solution that is also routable outside the convex boundary.*

PROOF. Here, we need to prove that there is a one-to-one correspondence between the maximal flow in $\mathcal{N}_F$ and the maximal escape routing solution that is also routable outside the convex boundary. First, we will prove that any valid flow solution in $\mathcal{N}_F$ corresponds to a valid escape routing solution. We can state that any valid flow solution in $\mathcal{N}_F$ must satisfy all the conditions defined in Section 3, since $\mathcal{N}_C$ models the constraint forest exactly. We have also shown in Section 3 that these constraints are *sufficient* for routability outside the convex boundary. Hence, there is a valid escape routing solution corresponding to any flow in $\mathcal{N}_F$. Then, we can prove that there is a flow in $\mathcal{N}_F$ corresponding to any valid escape routing solution. We have proven in Section 3 that the constraints defined are *necessary* for routability outside the convex boundary. So, any valid escape routing solution must satisfy all these constraints; hence must have a corresponding valid flow in $\mathcal{N}_F$. □

Table 1: Comparison of routability-driven escape routing with the traditional algorithm

PIN CLUSTER		TRADITIONAL ESCAPE ROUTING		ROUTABILITY-DRIVEN ESCAPE ROUTING	
Area	# Pins	# layers	time	# layers	time
7167	1687	13	1:12	10	0:57
8530	2080	14	1:44	11	1:26
9237	3742	26	3:34	21	2:53
9930	4885	31	5:02	26	4:13
10620	5984	38	7:04	32	5:51
12534	7638	47	11:02	40	8:59

Table 2: Single-layer routing characteristics of the traditional and routability-driven escape routing algorithms

PIN CLUSTER		TRADITIONAL ESCAPE ROUTING			ROUTABILITY-DRIVEN ESCAPE ROUTING		
Area	# Pins	# escape	# routable	time	# escape	# routable	time
7167	1687	308	239	0:14	290	290	0:13
8530	2080	322	264	0:17	310	310	0:17
9237	3742	329	270	0:21	319	319	0:21
9930	4885	337	287	0:24	331	331	0:24
10620	5984	344	285	0:27	333	333	0:26
12534	7638	368	299	0:35	353	353	0:34

6. EXPERIMENTAL RESULTS

We have performed experiments to evaluate the practical effectiveness of the models and algorithms we have proposed. We have implemented all algorithms in C++, and performed the experiments on a Linux system with Intel Centrino 1.5GHz processor, and 512MB memory.

For comparison purposes, we have applied a network flow based escape routing algorithm on a set of test circuits, and then used the optimal algorithm (proposed in Section 4) to select the maximal subset of routes that are also routable outside the boundary. In other words, escape routing is performed without considering routability outside in the beginning, and then the unroutable nets for the current layer are removed. The results of this methodology are given in Tables 1 and 2 under the columns *traditional escape routing*. We have also implemented the integrated approach (proposed in Section 5), which considers routability constraints outside the boundary during the actual escape routing algorithm. A sample routing solution using this integrated methodology is given in Figure 15. Note here that all the necessary and sufficient conditions defined in Section 3 are satisfied in this solution, and it is guaranteed that all nets can be routed outside without any conflicts.

Table 1 gives the final routing results corresponding to these two methodologies. When routability outside the boundary is not considered during the actual escape routing, more routing layers

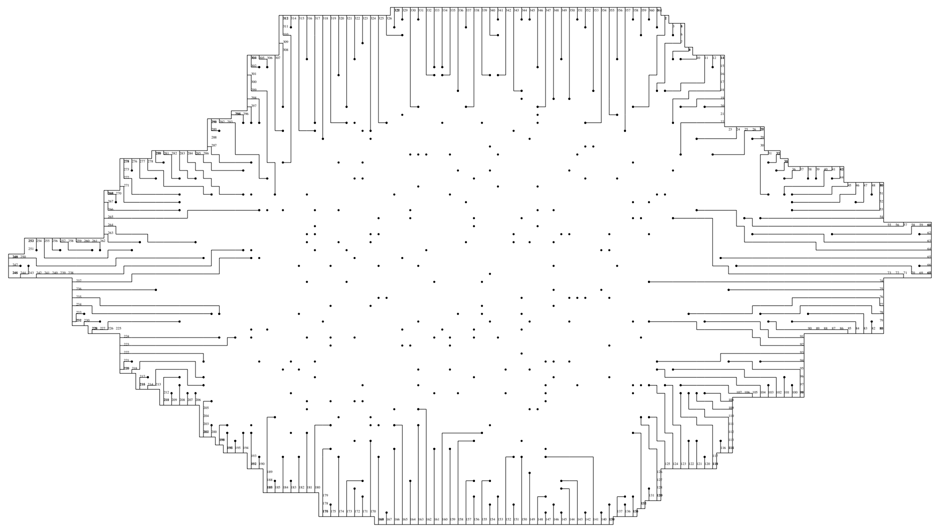

Figure 15: The escape routing solution on one layer of a sample pin cluster. 227 out of 414 nets have been routed on this layer. The solution found is also guaranteed to be routable outside.

are needed, as can be observed in this table. However, when these constraints are integrated into the actual escape routing algorithm, the number of necessary layers decreases by 17% on the average.

We have also listed the number of nets routed on the first layers of each circuit in Table 2. These quantities are important to observe the characteristics of these algorithms more closely, because each algorithm tries to route the maximal number of nets on the first layer. In this table, we list not only the number of nets that have escaped to the convex boundary, but also the number of nets that are routable outside. On the average, the traditional escape routing algorithm routes 3.7% more nets on the first layer. However, on average 18.1% of these nets are not routable outside the boundary; so they need to be removed from the solution of this layer (and need to be propagated to the lower layers). However, when we consider routability constraints during the actual escape routing algorithm, it is guaranteed that the escape routing solution found is completely routable outside.

7. CONCLUDING REMARKS

In this paper, we have studied the escape routing problem of irregular-shaped pin clusters, which are encountered frequently in in high-end MCMs. We have shown that routing nets to the cluster boundary without considering routability outside may lead to inferior solutions. We have proposed a set of necessary and sufficient conditions that model routability based on the positions of escape terminals on the boundary. Then, we have proposed an algorithm that selects the optimal subset of escape routes that are also routable outside. This algorithm is especially useful when a traditional routing algorithm is applied on a cluster of pins with a convex boundary. Then, we have shown how to integrate these constraints into the original routing algorithm without losing optimality. Our experiments have shown that the integrated methodology can reduce the number of layers by 17% on the average.

8. REFERENCES

[1] V. Betz and J. Rose. VPR: A new packing, placement and routing tool for FPGA research. In *Proc. of the 7th International Workshop on Field-Programmable Logic*, pages 213–222, 1997.

[2] G. Capwell. High density design with MicroStar BGAs. Application Report SPRA471A, Texas Instruments, 1998.

[3] W.-T. Chan, F. Y. L. Chin, and H.-F. Ting. Escaping a grid by edge-disjoint paths. In *Proc. of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 726–734, 2000.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1992.

[5] C. Ebeling, L. McMurchie, S. A. Hauck, and S. Burns. Placement and routing tools for the triptych fpga. *IEEE Trans. on VLSI*, pages 473–482, 1995.

[6] J. Hershberger and S. Suri. Efficient breakout routing in printed circuit boards. In *Proc. of the thirteenth annual symposium on Computational geometry*, pages 460–462, 1997.

[7] G. A. Katopis, W. D. Becker, T. R. Mazzawy, H. H. Smith, C. K. Vakirtzis, S. A. Kuppinger, B. Singh, P. C. Lin, J. Bartells, G. V. Kihlmire, P. N. Venkatachalam, H. I. Stoller, and J. L. Frankel. MCM technology and design for the S/390 G5 system. *IBM J. of Research and Development*, 43, 1999.

[8] J. Ludwig. IBM Systems Group. Private communication, 2004.

[9] R. Nair. A simple yet effective technique for global wiring. *IEEE Trans. on Computer-Aided Design*, 6, 1987.

[10] A. Titus, B. Jaiswal, T. Dishongh, and A. N. Cartwright. Innovative circuit board level routing designs for bga packages. *IEEE Trans. on Advanced Packaging*, 27, 2004.

[11] D. Wiens. Printed circuit board routing at the threshold. *The Board Authority*, pages 44–47, December 2000.

[12] M. Yu and W. W. Dai. Single-layer fanout routing and routability analysis for ball grid arrays. In *Proc. of Intl. Conf. on Computer-Aided Design (ICCAD)*, pages 581–586, 1995.