# On Interactions Between Routing and Detailed Placement

**Devang Jariwala**
University of Illinois at Chicago
Dept. of Computer Science
Chicago, IL 60607
djariwal@cs.uic.edu

**John Lillis**
University of Illinois at Chicago
Dept. of Computer Science
Chicago, IL 60607
jlillis@cs.uic.edu

## Abstract

*The main goal of this paper is to develop deeper insights into viable placement-level optimization of routing. Two primary contributions are made. First, an experimental framework in which the viability of* predictive *models of routing congestion for optimization during detailed placement can be evaluated, is developed. The main criteria of consideration in these experiments is how (un)reliably various models from the literature detect routing hot-spots. We conclude that such models appear to be too unreliable for detailed placement optimization. Second, motivated by the first result, we present a unified combinatorial framework in which cell placement and* exact *routing structures are captured and optimized; the framework relies on the* trunk-decomposition *of global routing structures and optimization is performed by a generalized* optimal interleaving *algorithm [5]. A proof of concept implementation of this framework is studied in the FPGA domain. The technique can reduce the number of channels at maximum density by almost 45% on average with maximum reduction of 68% for optimized global routing.*

## 1 INTRODUCTION

It has been long understood that the traditional serialized design flow of digital systems – e.g., logic synthesis followed by floorplanning, followed by placement and routing – introduces potential *optimization disconnects*: optimization decisions made during one stage may not correlate well with solution quality at subsequent stages. This paper investigates this issue at the boundary between *detailed placement* and *global routing*.

### 1.1 Related Work

A number of prior works have examined the relationships between placement and routing (e.g., [4], [6], [7], [8], [9], [10], [12], [13]). Many of these works target directly or indirectly the issue of *white-space allocation* (e.g., [2], [3], [12], [9]). In general, these approaches exploit the fact that often significant portions of the layout area are unused by logic components and that by spreading out the components intelligently, regional routing demand can be better matched to routing supply. Some approaches employ probabilistic routing models to predict this demand (e.g. [2], [13], [10]) while others invoke a quick global router on a (not necessarily legal) placement (e.g., [9]).

Such methods focusing on white space allocation are fundamentally *global* in nature in that they pay little attention to, for example, the impact of the exact locations of cells within rows on routing. As such, the topic of this paper – interactions between detailed placement and routing – may be considered complementary to white space allocation methods.

More directly relevant to this paper are works related to *congestion prediction* (e.g., [4], [7], [6], [13]). Generally speaking, predictive techniques take a cell placement and estimate or predict routing congestion. Attempts have been made to integrate such models into placement objectives (e.g., [4]) and other papers have focused evaluating the predictive capability of the methods (e.g., [6]) with no integration into an optimization engine.

Additional related work includes that of [8] which proposes tight integration of routing and placement operations in a simulated annealing framework targeting FPGAs by embedding a global router inside the simulated annealing loop. The router is invoked each time a move is made to rip-up and reroute parts of affected nets. Thus, the approach achieves integration between placement and routing by *alternating* between traditional placement and routing perturbations.

### 1.2 Contributions

The results presented herein are of two types. First, we evaluate the potential of predictive models for routability optimization during detailed placement. The goal is to determine if it is indeed worthwhile to pursue this avenue. An experimental framework is presented in which we may evaluate how well a model predicts routing hot-spots in the layout. From these experiments we draw the conclusion that such models appear to be too unreliable for our purposes.

Second, given the conclusions of the preliminary experiments, we develop a framework in which cells and *exact* global routing structures are captured in a unified combinatorial model. At the core of the framework is what we call a *trunk decomposition* of global routes which, essentially, allows us to view wire segments as "placeable objects." Using this framework, we develop an optimization engine based on a generalization of *optimal interleaving* [5] which simultaneously optimizes placement of cells and wiring segments. Thus, while the goal is similar to that of [8] (tight integration of routing and detailed placement), the approach is a substantial departure from prior work in how the solution space is viewed, represented and optimized. Experimental results indicate the promise of the approach.

The remainder of the the paper is organized as follows: next we describe experimental setup on applicability of *predictive* approach to routability optmization followed by our

proposed *exact* alternative in Section 3. Section 4 discusses the experimental results and we draw conclusions in Section 5.

## 2 SUITABILITY OF ESTIMATION TECHNIQUES

As stated in the introduction, our main interest is in congestion optimization during detailed placement. Fittingly, when first embarking on this research topic, we examined techniques for modeling or predicting congestion (e.g., [4], [7], [6]) with the idea of incorporating such models into a detailed placement engine.

In order for this strategy to be successful, at a minimum two criteria should be met:

(1) The model must effectively predict hot spots among the routing regions. It may not be necessary to be extremely accurate with respect to actual routing density values, but the *relative* density values between regions should exhibit high *fidelity* – i.e., if the model says one region is more congested than another, this must be reflected in an actual routing of the design (at least a high percentage of the time). It is clearly not sufficient if the model merely gives a reasonable prediction of the maximum overall density if the local information is not reliable.

(2) There should be a reasonable way to incorporate the model into an optimization engine.

If a model cannot satisfy the first requirement, the second becomes moot.

### Experiments

We have implemented three estimators available in literature: RISA [4], the combinatorial estimator of Lou et.al. [7] and fGREP [6]. The experiments were conducted to test the applicability of the models to detailed placement level optimization both in standard cell and FPGA domains.

For the standard cell case, IBM benchmarks [14] were used. Labyrinth [14] was used to route the placements obtained by Dragon placement tool [11]. We implemented the RISA model to estimate the routing demand for circuits placed by Dragon. Also, the combinatorial model of Lou et. al. was used for estimation.

For the FPGA case, MCNC benchmarks were placed and globally routed using the VPR tool [1]. In this case, we implemented fGREP [6] to estimate the routing demand for circuits placed by VPR. We also used RISA for routing demand estimation.

Within this setup we first performed experiments to evaluate the *accuracy* of the models with respect to maximum routing demand versus actual post-routing demand. In a second set of experiments we evaluate the *fidelity* of the models. Results are reported below.

#### *Results and Discussion*

First, we discuss the accuracy of the estimation models. This essentially means finding an answer to the following question: is the maximum demand predicted by the estimation model indeed the actual maximum demand according to router?

Table 1 shows the maximum routing densities for four different IBM benchmarks. For each of the models, we compute the maximum horizontal routing demand ($X_{max}$) and vertical routing demand ($Y_{max}$) among global routing tiles. We also run the Labyrinth router and get the actual maximum routing demand (both horizontal and vertical). As we can see from the table, the maximum demand predicted by the estimators is not close to the actual maximum routing demand. So we conclude that both estimators can *not* accurately predict the maximum routing demand.

**Table 1. Comparison of actual and estimated routing demand in standard cell domain. Grid size is the dimension of routing grid for Labyrinth global router. $D_{actual}$, $D_{risa}$ and $D_{comb}$ are the maximum global routing density according to Labyrinth, RISA and combinatorial model respectively.**

| Circuit | Grid Size | $D_{actual}$ | | $D_{risa}$ | | $D_{comb}$ | |
|---|---|---|---|---|---|---|---|
| | | $X_{max}$ | $Y_{max}$ | $X_{max}$ | $Y_{max}$ | $X_{max}$ | $Y_{max}$ |
| IBM 01 | 64x64 | 16 | 15.5 | 5.17 | 3.76 | 5.09 | 4.67 |
| IBM 02 | 80x64 | 36 | 25 | 56.70 | 34.34 | 48.98 | 42.37 |
| IBM 03 | 80x64 | 30.5 | 22 | 45.13 | 36.08 | 41.30 | 36.26 |
| IBM 04 | 96x64 | 26 | 24 | 39.33 | 32.77 | 43.81 | 40.56 |

For the FPGA domain, similar data is shown for some MCNC benchmarks in Table 2. In this case, the routing fabric is composed of routing channels in X and Y direction. The fGREP estimator, designed specifically to work on FPGAs, works well with this configuration. But RISA uses a tile based approach to predict global routing demand. Hence, in case of RISA, square routing tiles are used. Each tile spans two logic blocks in a row and two logic blocks in a column. Thus, the actual routing demands are computed accordingly for each tile. The maximum routing demand for fGREP and RISA are shown in Table 2. The maximum estimated demand is not close to the actual routing demand (by VPR). And in this case also, both estimators can *not* accurately predict the actual maximum routing demand.

**Table 2. Comparison of actual and estimated routing demand in FPGA domain. $D_{actual}$, $D_{fgrep}$ and $D_{risa}$ are maximum global routing demand according to VPR, fGREP and RISA respectively.**

| Circuit | $D_{actual}$ | | $D_{fgrep}$ | | $D_{actual}$(2x2) | | $D_{risa}$(2x2) | |
|---|---|---|---|---|---|---|---|---|
| | $X_{max}$ | $Y_{max}$ | $X_{max}$ | $Y_{max}$ | $X_{max}$ | $Y_{max}$ | $X_{max}$ | $Y_{max}$ |
| alu4 | 7 | 7 | 9.58 | 9.13 | 28 | 28 | 16.85 | 15.72 |
| apex2 | 8 | 8 | 11.01 | 10.44 | 32 | 32 | 20.63 | 21.88 |
| apex4 | 9 | 9 | 11.29 | 11.97 | 36 | 36 | 23.25 | 23.63 |
| bigkey | 5 | 5 | 8.45 | 7.35 | 20 | 20 | 10.15 | 8.38 |

From the above we conclude that the accuracy of the models in predicting maximum demand is very limited. However, this alone does not necessarily imply that they cannot be used effectively for optimization – i.e., low accuracy can be tolerated so long as the models demonstrate high *fidelity* with the

actual data. This means, the maximum routing demand according to model may not be close to the absolute maximum routing demand according to the router, but if the estimator can faithfully identify the routing hot-spots, it may be helpful in improving routability.

To further study the fidelity of estimation, we first concentrate on circuit IBM 02. The first part of Table 3 presents the distribution of the global routing tiles according to the Labyrinth router and the RISA estimator. Each column in the table shows number of tiles by percentage of horizontal demand ($D_x$) with respect to maximum horizontal demand ($X_{max}$) according to corresponding model. The granularity is finer toward higher demand, since these tiles are critical in routability optimization. It can be observed that the number of tiles with routing demand greater than 95% are very different in both cases. Moreover, another interesting aspect would be how many of the tiles, which are indicated as critical by estimation model, are actually critical according to router.

**Table 3. Distribution of tiles by $D_x$ for IBM 02 circuit**

| Model | # Tiles by $D_x$ percentage of $X_{max}$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 97%-100% | 95%-97% | 90%-95% | 80%-90% | 70%-80% | 50%-70% | 0%-50% |
| Laby. | 27 | 89 | 1832 | 658 | 247 | 517 | 1750 |
| RISA | 2 | 3 | 4 | 58 | 306 | 936 | 3811 |

Figure 1 (a) shows the distribution of the tiles having $D_{x\_actual}$ greater than 95% of $X_{max\_actual}$ (there are 116 tiles for IBM 02 circuit). As it is evident from the data, none of these tiles have $D_{x\_risa}$ greater than 90% of the $X_{max\_risa}$, i.e. these are false negative indication of routing hotspots. Similarly, the distribution of critical tiles having $D_{x\_risa}$ greater than 95% of $X_{max\_risa}$ is shown in the Figure 1 (b). All these tiles have $D_{x\_actual}$ between 90% and 95% of $X_{max\_actual}$. So we can conclude that the model does not exhibit required *fidelity*. The data for the combinatorial model is also similar.

Corresponding data for the FPGA case is shown in Table 4 and Figure 2 for apex4 circuit. Similar conclusions can be drawn here as well. In thise case, fGREP identifies some channels as near ctitical though actually they are not, i.e. there are false positive indications.

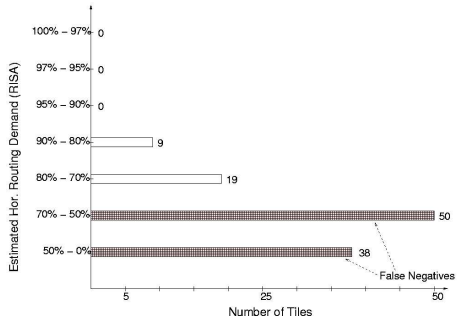
**Table 4. Distribution of channels by $D_x$ for apex4 circuit**

| Model | # X channels by $D_x$ as a percentage of $X_{max}$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 97%-100% | 95%-97% | 90%-95% | 80%-90% | 70%-80% | 50%-70% | 0%-50% |
| VPR | 297 | 0 | 0 | 246 | 224 | 265 | 412 |
| fGREP | 6 | 4 | 34 | 188 | 235 | 530 | 447 |

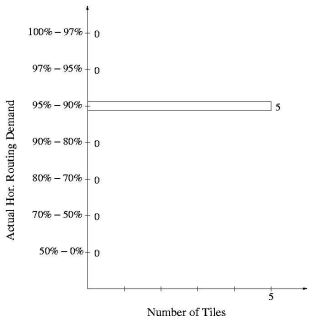
As it can be seen, the models studied here have limited fidelity in predicting routing demand. Hence, we conclude that they are not suitable for building a robust detailed placement-level routability optimization engine.

## 3 A NON-PREDICTIVE APPROACH

In light of the conclusions drawn in the previous section, in this section we present a non-predictive, exact approach




(a) Distribution of tiles with $D_{x\_actual} > 0.95 \times X_{max\_actual}$




(b) Distribution of tiles with $D_{x\_risa} > 0.95 \times X_{max\_risa}$

**Figure 1. Distribution of tiles by $D_x$ for IBM 02 circuit**

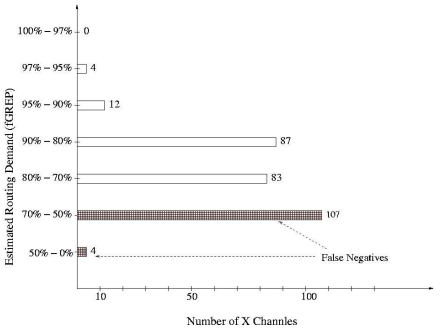
to optimize routability at detailed placement-level. The placement-level routability optimization problem can be formulated as: Given a circuit, i.e. a set of cells **C** and a set of nets **N**, an initial placement and global routing, we want to find placement and routing such that routing congestion is minimized.

The proposed approach is a generalization of optimal interleaving technique introduced in [5]. So we first give an overview of the optimal interleaving technique. Then we propose a generalization of this technique to address *exact* routing congestion.

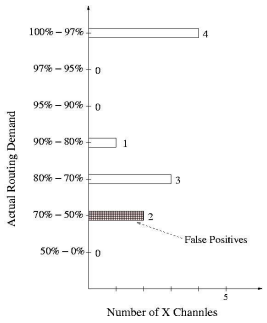
### 3.1 Interleaving

Interleaving is a wirelength driven, detailed placement optimization technique that was introduced in Mongrel, a standard cell placement tool [5]. It is a powerful intra-row optimization technique. The basic technique is described here for reference.

Interleaving considers a part of a row, called window $W$, for optimization. A window $W$ is essentially a sequence of cells in a row of detailed placement. The technique partitions the window into two disjoint sequences **A** and **B**, keeping the original order of the cells in each sequence, i.e. a cell precedes another in sequence **A** (**B**) if and only if the former precedes the later in original sequence $W$. The algorithm then finds the *optimal* interleaving of these two sequences **A** and **B** to optimize the wirelength. During the interleaving the relative order of the cells in each sequence is preserved, i.e. cell $a_i$ is always placed in interleaved sequence after cell $a_{i-1}$.




(a) Distribution of channels with $D_{x\_actual} > 0.95 \times X_{max\_actual}$




(b) Distribution of channels with $D_{x\_fgrep} > 0.95 \times X_{max\_fgrep}$

**Figure 2. Distribution of channels by $D_x$ for apex4 circuit**

It employs a dynamic programming algorithm to solve the interleaving problem. To formulate the recurrence, we define following terms: $W$ is a sequence of $c$ cells in a row, which is partitioned into two sequences: sequence **A**, with $m$ cells $(a_1, \ldots, a_m)$ and sequence **B** with $n$ cells $(b_1, \ldots, b_n)$. We define $S_{i,j}$ as the arrangement of cells $a_1, \ldots, a_i$ and cells $b_1, \ldots, b_j$ and cost of such arrangement is $C(S_{i,j})$. Let $S_{i,j}b_{j+1}$ denote the placement of the cell $b_{j+1}$ at the end of sequence $S_{i,j}$.

Having defined these terms, we are ready to write the recurrence for Dynamic Programming (DP):

$$\begin{aligned} S_{0,0} &= \phi \\ C(S_{0,0}) &= 0 \\ C(S_{i,j}) &= \begin{cases} C(S_{i-1,j}\ a_i) & \text{if } C(S_{i-1,j}\ a_i) \le C(S_{i,j-1}\ b_j) \\ C(S_{i,j-1}\ b_j) & \text{otherwise} \end{cases} \end{aligned} \quad (1)$$

The intuition behind the recurrence can be described as follows. We would like to find optimal configuration of first $i$ cells from sequence **A** and first $j$ cells from sequence **B**. So there are two ways this can be achieved: first, cell $a_i$ is at the end or second, cell $b_j$ is at the end. In the first case, the cell $a_i$ is appended at the end, hence all the cells preceding $a_i$ in sequence **A** and all the $j$ cells in sequence **B** have been placed, i.e. the configuration is $S_{i-1,j}a_i$. Similarly, for the second alternative, we have configuration $S_{i,j-1}b_j$. We take the better of the two alternatives.

Intuitively, the reason that dynamic-programming can be applied is that the subpoblems exhibit what might be called "separability." In particular, if we separate a sequence of cells by a vertical line, the minimum wire-length ordering of the left group of cells is independent of the ordering of the right group (because the nets crossing the line is independent of these orderings). Interleaving explores exponential solution space in polynomial time and original configuration is always in solution space.

### 3.2 Routing Based Interleaving

In this subsection, we present the details of the Routing Based Interleaving (RBI). As interleaving is a powerful technique for optimizing detailed placement, it is natural to try to extend it to tackle exact routing information. We can give a broad definition of interleaving in which it takes a sequence of objects and it gives us new permutation of the objects. Note that these objects need *not* be homogeneous. So if we can represent routing information with a set of "placeable objects," we can apply the interleaving with routing. In order to do that we decompose the routing topology of a net. The details are presented with island style architecture for FPGA.

#### *Trunk Decomposition*

To induce "placeable" routing objects, we propose to decompose routing topologies into entities of two types:

**Horizontal Trunk** A horizontal trunk of routing topology of a net is a longest continuous horizontal segment of topology.

**Vertical Segment** A vertical segment of the routing topology of a net is a continuous vertical component that is connected to at most two trunks, one at the top and the other at the bottom and is connected to at most one cell.

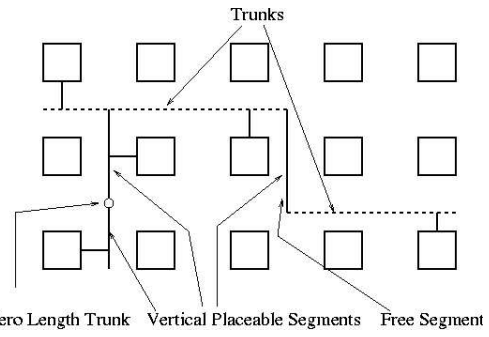



**Figure 3. Trunk Decomposition**

So the routing topology of a net is composed of two sets: a set of trunks ($T$) and a set of vertical objects ($O$). Together, they exactly define the global routing of the net i.e. there is one to one correspondance between these sets and global routing of a net. Figure 3 shows one such decomposition of net topology. Also any horizontal movement of the vertical segment, accompanied by contraction or expansion of the connected trunks, denotes a change in the routing topology of the corresponding net. Similarly, the horizontal segments can expand or contract when a cell connected to the same is

displaced in the same row. The vertical segments can also be connected to cells. These segments must move with connected cells to maintain electrical connectivity. The vertical segments which are not connected to any cells are called *free* segments.

### DP Formulation

In this section, we describe the proposed extension of the traditional optimal interleaving. In the case of RBI, the interleaving takes into account not only the cells but also the vertical segments from the trunk decomposition of routing topology of nets. RBI treats them as cells with zero width. The window of interleaving, $W$, is no longer just a sequence of cells, but a sequence of cells and a set of free vertical segments between two adjacent cells. The other difference is that the window is no longer restricted to a row, but as the vertical components can possibly have nonzero span, it is a rectangular portion of the placeable region.

The sequence of cells and objects, $W$, is partitioned into two disjoint sequences:

$$\begin{array}{lll} \mathbf{A} : (a_i, O_i) & 0 < i \leq m & \text{where } O_i \text{ is an ordered set} \\ & & \text{of free vertical segments} \\ \mathbf{B} : (b_j, P_j) & 0 < j \leq n & \text{where } P_j \text{ is an ordered set} \\ & & \text{of free vertical segments} \end{array}$$

As in the case of the traditional optimal interleaving, both the sequences keep the original relative order of cells and segments, i.e. cell $a_i$ is always after cell $a_{i-1}$ in the original sequence and also it follows all the vertical segments in set $O_{i-1}$. The same also holds for the vertical segments, i.e. all the segments in set $O_i$ are preceded by the cell $a_i$. It should be noted that any of the sets of the vertical segements can possibly be empty. Figure 4 shows one such partitioning of window $W$ into two sequences.

RBI maintains the relative order of cells and vertical segments of both the sequences in the final solution, i.e. cell $a_i$ is always placed only after cell $a_{i-1}$ and all the vertical segments in set $O_{i-1}$ have been placed. Moreover, any vertical segment in set $O_i$ is placed only after cell $a_i$ has been placed.

Let $S_{i,j,k,l}$ be the configuration in which all the cells $a_1, \ldots, a_i$ and $b_1, \ldots, b_j$ have been placed. Also, first $k$ vertical segments from set $O_i$ and first $l$ segments from set $P_j$ have been placed. Let the cost of this configuration be $C(S_{i,j,k,l})$. Let $S_{i-1,j,k,l}a_i$ denote placement of cell $a_i$ at the end of $S_{i-1,j,k,l}$, cost of which is $C(S_{i-1,j,k,l}) + C(a_i)$, where cost $C(a_i)$ also includes the cost of placing connected segments of the cell $a_i$. From now on, cost of placing a cell implicitly includes the cost of placing vertical segments connected to it. Let $S_{i,j,k,l}O_{i_{k+1}}$ denote placement of vertical segment $O_{i_{k+1}}$ at the end of configuration $S_{i,j,k,l}$ and cost of such configuration be $C(S_{i,j,k,l}) + C(O_{i_{k+1}})$. We can now write the DP formulation:

$$\begin{array}{lcl} S_{0,0,0,0} & = & \phi \\ C(S_{0,0,0,0}) & = & 0 \\ C(S_{i,j,k,l}) & = & min(C_A, C_B) \end{array} \qquad (2)$$

where,

$$\begin{array}{ll} C_A = min_{0 \leq x \leq l} & \{\, C(S_{i-1,j,|O_{i-1}|,x}) + C(a_i) \\ & + C(O_{i_1}, \ldots, O_{i_k}) + C(P_{j_{x+1}}, \ldots, P_{j_l}) \,\} \\ C_B = min_{0 \leq y \leq k} & \{\, C(S_{i,j-1,y,|P_{j-1}|}) + C(b_j) \\ & + C(O_{i_{y+1}}, \ldots, O_{i_k}) + C(P_{j_1}, \ldots, P_{j_l}) \,\} \end{array}$$

The intuition behind the above formulation is as follows. Suppose we want to find optimal configuration of first $i$ cells from sequence **A**, first $k$ vertical segments in set $O_i$, first $j$ cells from sequence **B** and first $l$ vertical segments in set $P_j$, i.e. configuration $S_{i,j,k,l}$. As in traditional optimal interleaving, we have two alternatives, either cell $a_i$ or cell $b_j$ is the last cell to be placed. Let's consider the first alternative. In this case, all the cells and segments preceding cell $a_i$ in sequence **A** must have been placed and all the cells upto and including cell $b_j$ in sequence **B** must also have been placed. Note that we can not place any segments from set $O_i$ before $a_i$ in order to maintain the relative order in each sequence. And hence all the $k$ segments of set $O_i$ must be placed after $a_i$. But a part of the first $l$ segments in set $P_j$ may be placed before cell $a_i$ but after cell $b_j$. So there is a range of choices available here: on one hand each of the $l$ vertical segments is placed after cell $a_i$ and on the other hand each of the $l$ vertical segments is placed before cell $a_i$. And all these constitute different configurations with different costs. We choose the configuration with the minimum cost as the best configuration where cell $a_i$ is last cell to be placed. We denote the cost of this configuration as $C_A$. Similarly, we choose the best configuration with cell $b_j$ as the last cell to be placed. For configuration $S_{i,j,k,l}$, we choose one of these having lower cost.

With respect to the correctness of the algorithm, we need a separability property as in the case of traditional optimal interleaving. Let us assume for the moment, that the objective function is simply maximum routing density. As in the case of traditional optimal interleaving, we observe that, for a particular partition of the components into left and right groups, minimization of routing density on the left hand side is independent of the ordering on the right hand side and vice-versa.

Figure 4 shows one example of RBI. For simplicity not all connections are shown. The RBI window contains four cells and some free segments. They are partitioned into two sequences. Cells $A$, $I$ and $N$ belong to sequence **A** and cell $E$ is in sequence **B**. The segments $\{B,\ C,\ D,\ G\}$ form ordered set $O_1$. Similarly, sets $O_2$, $O_3$ and $P_1$ are formed. A possible output of RBI is also shown. Note that cells $I$ and $E$ are swapped as a result of RBI. The vertical segment $G$ "jumps over" cell $E$, and is placed before cell $E$. Similarly, vertical segment $M$ "jumps over" cell $N$.

Figure 5 shows some of the possible displacements of cells and segments that can lead to reduction in density. Figure 5(a) shows how the displacement of a cell can lead to reduction in the density due to simultaneous displacement of a connected vertical segment. This displacement can reduce both the density in channels through which the trunk was

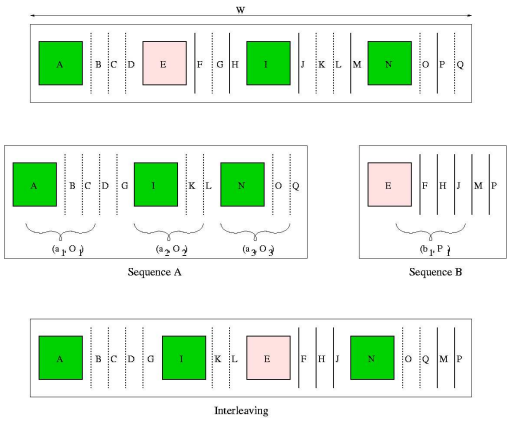



**Figure 4. Example of Routing Based Interleaving**

passing and wirelength. Figure 5(b) illustrates the possible displacement of a free vertical segment. The trunk connected to the bottom of the segment is passing through relatively high density channels, whereas the corresponding channels in the path of the top trunk of the segment are relatively less congested. In such cases, the displacement of the segment during RBI can lead to expansion of the top trunk through less congested area with the advantage of eliminating path through relatively highly congested area. Figure 5(c) is an example of simultaneous displacement of two cells toward each other. These cells have connected segments that are connected to a common trunk. The movement of the cells can lead to *merging* of the connected segments and corresponding trunk getting contracted to zero length. Figure 5(d) shows a possible outcome in which the congestion can be reduced at the cost of increase in wirelength. A vertical segment is passing through a channel with relatively high congestion. So possible displacement of segment, along with the connected cell, can reduce the congestion. But the zero-length trunk connected to the top of the segment expands (in low congestion channels) to increase the wirelength.

### *Cost Function*

The cost function for RBI is a key part. We want to minimize the maximum routing demand in order to maximize the routability. But in case of FPGAs, there can be a very large number of different solutions that have the same maximum routing density. So we need a better objective that can guide us in the right direction. We use a lexicographic combination of maximum density ($D_{max}$), number of channels at maximum density ($nMax$) and wirelength ($WL$) – i.e. $D_max$ is primary objective and $nMax$ and $WL$ serve as a primary and secondary tiebreakers respectively. This more powerful objective also observes the separability discussed earlier.

## 4 EXPERIMENTS

We have implemented RBI for the FPGA domain and performed initial experiments to test its effectiveness. The experiments were performed in Linux environment on a PC with 2.4GHz Intel Pentium 4 CPU and 512 MB RAM. The

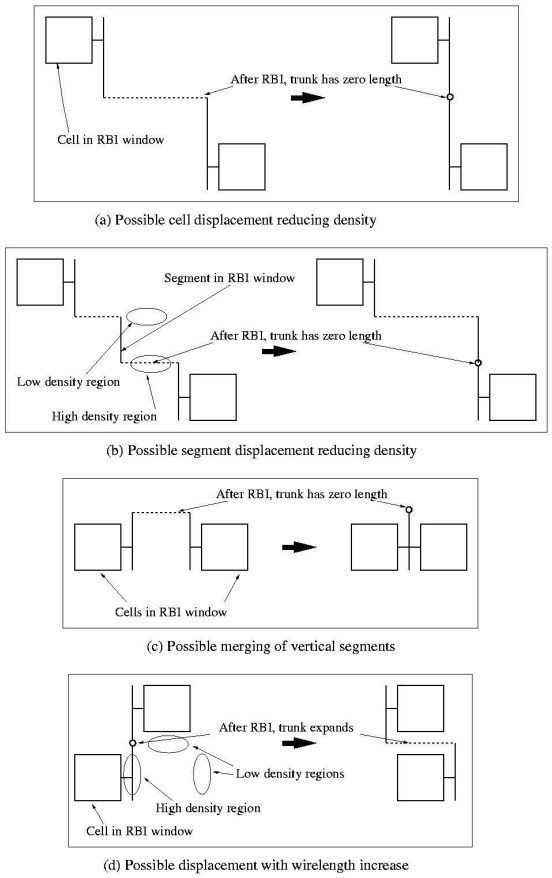



**Figure 5. Possible outcomes during RBI**

main criteria of comparison were maximum channel density ($D_{max}$), number of channels at maximum density ($nMax$) and wirelength ($WL$) of the routed circuit. All such statistics are reported in Table 5 for all 20 MCNC benchmark circuits. The placements were obtained with VPR placer running in wirelength driven mode. The placed circuit was routed with wirelength driven VPR global router to minimize number of tracks required to route the placement.

This placed and routed circuit data is given in first data-set (before RBI in Table 5). RBI was applied on them with the lexicographic cost objective discussed in previous section.

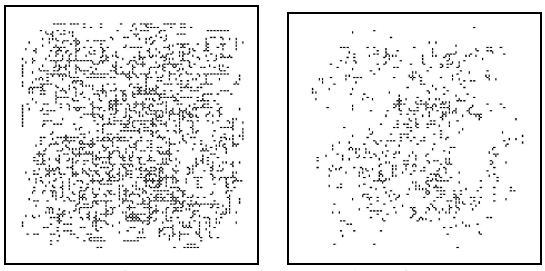
The formulation presented allows movement of cells and vertical segments only in horizontal direction. But the horizontal segments can only expand or contract, keeping their vertical positions unchanged. This may seem to be a limiting factor. But owing to the symmetry of the island style architecture, we can easily formulate the similar problem for vertical direction. In the experiments, RBI was applied both in horizontal and vertical directions.

As it can be observed, as a result of RBI application, the number of channels at maximum density is reduced by almost 45% on average. The best improvement in $nMax$ is 68.08% for ex1010 circuit. There is a minor increase of 1.04% in the wirelength compared to initial placement and routing with maximum increase less than 2% for all 20 benchmarks. The

**Table 5. Maximum Density ($D_{max}$), number of channels at maximum density ($nMax$) and wirelength ($WL$) results.**

| Circuit | Before RBI | | | After RBI | | | Change | | Time |
|---|---|---|---|---|---|---|---|---|---|
| | $D_{max}$ | $nMax$ | $WL$ | $D_{max}$ | $nMax$ | $WL$ | $nMax(\%)$ | $WL(\%)$ | |
| ex5p | 9 | 699 | 15180 | 9 | 466 | 15358 | -33.33% | 1.17% | 0.96 |
| tseng | 5 | 582 | 7787 | 5 | 425 | 7863 | -26.98% | 0.98% | 0.17 |
| apex4 | 9 | 628 | 16962 | 9 | 266 | 17119 | -57.64% | 0.93% | 1.08 |
| misex3 | 8 | 650 | 16900 | 8 | 256 | 17079 | -60.62% | 1.06% | 0.94 |
| alu4 | 7 | 916 | 16484 | 7 | 585 | 16591 | -36.14% | 0.65% | 0.62 |
| diffeq | 6 | 467 | 11970 | 6 | 178 | 12141 | -61.88% | 1.43% | 0.20 |
| dsip | 5 | 691 | 10991 | 5 | 435 | 11148 | -37.05% | 1.43% | 0.20 |
| seq | 8 | 1161 | 22547 | 8 | 689 | 22752 | -40.65% | 0.91% | 1.73 |
| apex2 | 8 | 1245 | 24332 | 8 | 770 | 24524 | -38.15% | 0.79% | 1.52 |
| s298 | 5 | 1720 | 16120 | 5 | 1370 | 16154 | -20.35% | 0.21% | 0.30 |
| des | 6 | 874 | 19190 | 6 | 403 | 19546 | -53.89% | 1.86% | 0.95 |
| bigkey | 5 | 1128 | 14046 | 5 | 819 | 14144 | -27.39% | 0.70% | 0.25 |
| frisc | 8 | 2202 | 43849 | 8 | 1518 | 44148 | -31.06% | 0.68% | 2.38 |
| spla | 9 | 2071 | 51127 | 9 | 1120 | 51590 | -45.92% | 0.91% | 2.60 |
| elliptic | 7 | 1881 | 36400 | 7 | 1367 | 36694 | -27.33% | 0.81% | 1.59 |
| ex1010 | 8 | 2027 | 55849 | 8 | 647 | 56581 | -68.08% | 1.31% | 2.25 |
| pdc | 11 | 2155 | 75642 | 11 | 883 | 76354 | -59.03% | 0.94% | 4.11 |
| s38417 | 6 | 1893 | 50622 | 6 | 713 | 51434 | -62.33% | 1.60% | 0.52 |
| s38584.1 | 6 | 1487 | 46975 | 6 | 483 | 47745 | -67.52% | 1.64% | 0.48 |
| clma | 8 | 4900 | 103832 | 8 | 2722 | 104712 | -44.45% | 0.85% | 2.70 |
| **Average** | | | | | | | **-44.99%** | **1.04%** | **1.27** |

congestion map for this run is shown before RBI in Figure 6 (a), where only channels at maximum density are shown in black. As it can be seen from Figure 6 (b), RBI reduces the congestion considerably. The runtime of RBI is given as a ratio of the RBI runtime to combined VPR placement and routing runtime. The average runtime is 1.27 times that of VPR flow.



(a) Before RBI (b) After RBI

**Figure 6. Congestion map for ex1010 circuit.**

## 5 CONCLUSIONS

In this paper, we first presented the experimental study of the various routability estimation techniques. The results of the studies indicate that such techniques can not faithfully identify the routing hot-spots. Hence, we conclude that they are not suitable for placement-level routability optimization.

We proposed an alternative, non-predictive, approach for *exact* routability optimization based on *trunk decomposition* and generalized *optimal interleaving*. The approach can optimize the placement of both cells and wiring segments simultaneously. It should be noted that *trunk decomposition* is a powerful technique which can be used to transform routing problem into placement problem and one can potentially explore routing problem in a different direction. Moreover, modularity of approach makes it easier to augment existing design flows.

A prototype targeting the FPGA domain was implemented and studied experimentally. The implementation presented incorporates maximum routing demand and wirelength in the cost objective. Experiments demonstrate the potential of the approach. Starting from an optimized global routing, the number of channels at maximum density can be reduced by almost 45% on average and 68% at maximum with nominal wirelength increase. Ongoing work includes algorithmic speed up techniques.

We argue that a similar approach can also be applied to standard and mixed cell domain for detailed routability optimization and future work will include this application.

## REFERENCES


[1] V.Betz, J.Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," *International Workshop on Field Programmable Logic and Applications*, 1997.

[2] U.Brenner, A.Rohe, "An effective Congestion Driven Placement Framework," *ISPD*, 2002.

[3] A.Caldwell, A.B.Kahng, I.L.Markov, "Hierarchical Whitespace Allocation in Top-down Placement," *IEEE TCAD*, vol. 22(11), November 2003.

[4] C.E.Cheng, "RISA: Accurate and Efficient Placement Routability Modeling," *ISPD*, 2001.

[5] S.W.Hur, J.Lillis, "Mongrel: Hybrid Techniques for Standard Cell Placement," *ICCAD*, 2000.

[6] P.Kannan, S.Balachandran, D.Bhatia, "On metrics for comparing routability estimation methods for FPGAs," *DAC*, 2002.

[7] J.Lou, S.Krishnamoorthy, H.Sheng, "Estimating Routing Congestion using Probabilistic Analysis," *ISPD*, 2001.

[8] S.Nag, R.Rutenbar, "Performance Driven Simultaneous Placement and Routing for FPGAs," *IEEE TCAD*, Vol 7, No 6, June 1998.

[9] P.Parakh, R.Brown, K.Sakallah, "Congestion Driven Quadratic Placement," *DAC*, 1998.

[10] M.Wang, X.Yang, K.Eguro, M.Sarrafzadeh, "Multi-center Congestion Estimation and Minimization During Placement," *ISPD*, 2000.

[11] M.Wang, X.Yang, M.Sarrafzadeh, "Dragon-2000: Standard-cell Placement Tool for Large Industry Circuits," *ICCAD*, 2000.

[12] X.Yang, B.Choi, M.Sarrafzadeh, "Routability Driven Whitespace Allocation for Fixed-die Standard-cell Placement," *ISPD*, 2002.

[13] X.Yang, R.Kastner, M.Sarrafzadeh, "Congestion Estimation During Top-down Placement," *ISPD*, 2001.

[14] `http://www.ece.ucsb.edu/~kastner/labyrinth`