GLARE: Global and Local Wiring Aware Routability Evaluation

Yaoguang Wei[1], Cliff Sze[2], Natarajan Viswanathan[3], Zhuo Li[2], Charles J. Alpert[2], Lakshmi Reddy[4], Andrew D. Huber[4], Gustavo E. Tellez[5], Douglas Keller[4], Sachin S. Sapatnekar[1]
[1]Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, USA
[2]IBM Austin Research Lab, Austin, TX, USA
[3]IBM Systems and Technology Group, Austin, TX, USA
[4]IBM Systems and Technology Group, Hopewell Junction, NY, USA
[5]IBM Systems and Technology Group, Burlington, VT, USA
Email: weiyg@umn.edu; {csze,nviswan,lizhuo,alpert,reddyl,adhuber,tellez,kellerd}@us.ibm.com; sachin@umn.edu

ABSTRACT

Industry routers are very complex and time consuming, and are becoming more so with the explosion in design rules and design for manufacturability requirements that multiply with each technology node. Global routing is just the first phase of a router and serves the dual purpose of (i) seeding the following phases of a router and (ii) evaluating whether the current design point is routable. Lately, it has become common to use a “light mode” version of the global router, similar to today’s academic routers, to quickly evaluate the routability of a given placement. This use model suffers from two primary weaknesses: (i) it does not adequately model the local routing resources, while the model is important to remove opens and shorts and eliminate DRC violations, (ii) the metrics used to represent congestion are non-intuitive and often fail to pinpoint the key issues that need to be addressed. This paper presents solutions to both issues, and empirically demonstrates that incorporating the proposed solutions within a global routing based congestion analyzer yields a more accurate view of design routability.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Integrated Circuit—*Design Aids - Routing*

General Terms

Algorithms, Design, Experimentation, Measurement

Keywords

Physical design, Routing, Routability evaluation, Local wiring modeling, Congestion metric

1. INTRODUCTION

Routability has become an increasingly important and difficult issue in nanometer-scale VLSI designs, and must be addressed across the entire physical synthesis tool stack. This in turn requires fast, yet reasonably accurate techniques to identify routing-challenged regions (hot spots), for routability optimization. This work focuses on the two key components of routability evaluation: (a) the method used to analyze the congestion of a given placement or design point, and (b) the metric(s) used to score or represent the congestion.

1.1 Congestion analysis techniques

Proposed methods to perform congestion analysis include:

1. Take a design through detailed routing to determine if it is routable or not.
2. Use a probabilistic congestion estimation procedure, without performing any routing [6, 13].
3. Perform fast global routing and use its solution to perform congestion analysis [10–12].

In principle, detailed routing estimates are the most accurate, but this approach is very time-consuming and impractical during the early stages of design closure. Probabilistic methods are highly inaccurate and fail to capture the behavior of global routing, especially in modern designs with numerous IP blockages, and a large number of metal layers with varying width and spacing. Lately, the third method has become more attractive and mainstream due to the advent of fast, high-quality global routers [3, 4, 7, 14, 15].

Although global routing based congestion analysis provides a happy medium between probabilistic analysis and detailed routing, it suffers from a key drawback: local congestion, or local routing resource usage is not accounted for during global routing. Local congestion clearly consumes varying amounts of routing resources depending on factors such as design rules, size of the global routing cell and pin density. As shown in Section 5, ignoring these effects can greatly mispredict design routability. Hence, local congestion needs to be modeled; and the method should be *flexible*, for it to be adjusted in a straightforward manner from one technology to the next (as design rules are different for each technology).

1.2 Metrics to score or represent congestion

Visual inspection of congestion plots (a region-wise color-coded map marking out the hot spots with the greatest contention for wiring resources) often serves as a first order method to compare the routability of different design points. However, optimization tools and designers also require a single metric that can accurately score or represent the design congestion.

Commonly used metrics in academia and industry are:

Overflow based metrics include total overflow (TOF) and maximal overflow (MOF) that measure the excess of the routing demand over routing capacity on the edges in a global routing graph (defined in Section 2). These metrics often fail to provide a clear picture of the design routability. For example, a global routing solution with an overflow of 700 might still be routable, as the overflow could be absorbed by neighboring regions or resolved during the subsequent routing phases. Further, their lack of intuition (e.g., how good/bad is an overflow of $14,253$?) makes it difficult to quantify how much better one design point is versus another.

Net congestion based metrics [2] include[1]: (a) $ACN(x)$, the average net congestion, defined as the average congestion of the top $x\%$ congested nets, where the congestion of a net is the maximum congestion among all the global routing edges traversed by the net. (b) $WCI(y)$, the worst congestion index, defined as the number of nets with congestion greater than or equal to $y\%$. In practice, $ACN(20)$, $WCI(90)$ and $WCI(100)$ are employed. The main issue with these metrics is that they fail to differentiate between a net spanning a single congested global routing edge and one that spans multiple congested edges.

In this paper, we propose to enhance the accuracy and effectiveness of routability evaluation. Our key contributions include:

[1]We name the metrics differently from [2] to facilitate later references.

- A study of the inaccuracies in existing global routing based congestion analyzers, specifically due to the lack of local routing resource modeling.
- An analysis of the weaknesses in existing metrics to score or represent design congestion.
- Methods to model and incorporate the effects of local routing resource usage during global routing, the impact of which is two-fold: (a) significant improvement in the accuracy of congestion analysis, (b) better prediction of detailed routing issues such as opens and shorts.
- A new congestion metric that is more intuitive and represents the design congestion with high fidelity.
- Detailed empirical validation of our proposed techniques on advanced industrial designs.

The rest of this paper is organized as follows. Background and definitions are presented in Section 2. Section 3 presents our methods for modeling local routing congestion. Section 4 describes our new metric for routability evaluation. Empirical validation and concluding remarks are provided in Sections 5 and 6, respectively.

2. PRELIMINARIES

Typically, during global routing, the chip is tessellated into $n_r \times n_c$ grids (or *g-cells*), and the global routing graph (GRG), $G = (V, E)$, is constructed. A node in V represents a g-cell in the layout, and an edge (called a *g-edge*) in E denotes the boundary between two adjacent g-cells. An example of the GRG is shown in Figure 1.

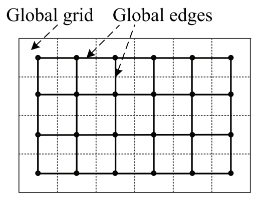


Figure 1: Global routing graph (GRG).

We now introduce some notation and terms that will be used in the remainder of this paper. For each edge e in the GRG, we define c_e as edge capacity – the total or maximal capacity of the edge, b_e as edge blockage, that needs to be discounted from c_e, and w_e as the routing demand on the edge. In global routing, c_e, b_e and w_e are generally expressed in the number of routing tracks, where a routing track is the routing resource taken by a single wire passing through an edge in the GRG. Let $o_e = \max(w_e + b_e - c_e, 0)$ be the overflow of an edge e. The total overflow of the layout is given by $\sum_{e \in E} o_e$, and the maximal overflow is given by $\max_{e \in E} o_e$. The congestion of edge e, denoted as g_e, is given by $g_e = (w_e + b_e)/c_e$.

3. MODELING OF LOCAL ROUTES

In this section, we first analyze the problems associated with existing congestion analysis methods, and then propose our model that captures the congestion due to local routes.

3.1 Limitations of existing global routing based methods

As mentioned in Section 1, global-routing based congestion analysis is now mainstream. Examples include, FastRoute [9] and NTHU-Route 2.0 [3], used as congestion analyzers within routability-driven placers IPR [10] and CRISP [11], respectively.

Global routers generally abstract the routing problem and only focus on g-cell-to-g-cell routes. In this case, they ignore the congestion due to local routes connecting the pins inside a g-cell. The problem is shown in Figure 2. The net (S, T) is a local net with two pins in a g-cell that are to the left of g-cell center b, and will definitely occupy some routing resources. However, due to the abstraction in global routers, the routing resources occupied or blocked by the local route connecting S to T are usually not modeled in congestion calculation.

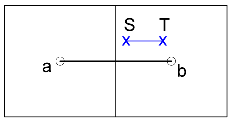


Figure 2: Local routes ignored by some global routers.

On the other hand, one of the major objectives of commercial global routers is to correlate with detailed routing. Designers expect global routers to report routing errors (usually in the form of congestion hot spots) without running detailed routing for hours or days. Therefore, industrial global routers have methods of varied sophistication to consider local routes. A simple approach is to reduce the g-cell size. Alternatively, some global routers include some form of detailed routing. However, these approaches to consider local routes greatly increase routing runtime and memory. When using global routers as congestion analyzers during physical synthesis, such routers are computationally expensive for tens or hundreds of invocations.

Our analysis in Section 5 shows that without considering local routes, a congestion analyzer significantly underestimates the congestion, and is unable to predict the problematic regions with opens and shorts in detailed routing. This motivates our work to introduce a simple and fast method to consider local routes when using a global router for congestion analysis.

3.2 Fast methods for modeling local routes

In this section, we present two methods to quickly model local routes: the first method estimates the local routes by the Steiner tree wirelength inside each g-cell, and the second method estimates the local routes based on the pin density of each g-cell.

3.2.1 Method 1: Estimation of local routes based on Steiner tree wirelength

In Figure 2, we observed that the longer the local route is, the more it would block the global routing track on that g-edge. This observation can be formulated by the following equation:

$$t_b = l_r / s_e, \tag{1}$$

where t_b is the number of routing tracks blocked by local route (S, T), l_r is the length of the local route, and s_e is the length of the g-edge. Note that this equation just serves as the basis for our method and will be extended in later use.

One may argue that if the detailed router somehow does not route net (S, T) along g-edge (a, b) (which is possible if the router uses a detour), blocking g-edge (a, b) leads to pessimistic congestion maps. However, we usually have no way to determine how a router would connect a pin until the routing is completed. In order to perform fast congestion analysis, it is critical to model local routes in a manner that is independent of the process of routing. Moreover, it is generally a good practice for congestion analysis to be pessimistic for the sake of routing closure.

We adopt equation (1) to calculate blocked tracks on a g-edge because we can easily extend it to include local routes when there are multiple pins covered by a g-cell. Consider the case when all the pins of a net are within a g-cell. To estimate local routing, we first build a Steiner tree (alternatively, one can use minimum spanning tree) for the pins. In our experiments, we use Flute [5]. We then break each horizontal tree segment into two based on the x-coordinate of the g-cell center and apply equation (1) to calculate blocked global routing tracks for the g-edges on each side of the g-cell boundaries. Similarly each vertical Steiner tree segment can be broken using the y-coordinate of the g-cell center.

An example is shown in Figure 3, where net (A, B) is a two-pin net while (A, J) and (B, J) are the two segments of a Steiner tree. The global routing tracks blocked by net (A, B) in the horizontal direction can be calculated based on segment (A, J). Since the g-cell center b is between A and J, segment (A, J) blocks global routing tracks on g-edges (a, b) and (b, c). The blocked tracks on g-edge (a, b) can be calculated as $(x_b - x_A)/(x_b - x_a)$, where x_b denotes the x-coordinate of g-cell center b, and other notations are defined similarly. Accordingly, the blocked global routing tracks on g-edge (b, c) is $(x_J - x_b)/(x_c - x_b)$. The vertical tracks blocked by net (A, B) can be calculated similarly, based on segment (J, B). As another example, when a segment is completely on the left of (above) or right of (below) the g-cell center, such as the net (C, D) in Figure 3, the blocked tracks can be calculated as $(x_D - x_C)/(x_c - x_b)$. In this case, only the tracks on g-edge (b, c) are blocked.

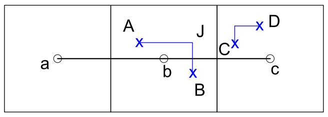


Figure 3: Local routing resource estimation for two-pin nets.

The proposed method can be easily applied to more complex Steiner trees, for example A, B, C in Figure 4. However, when a net has pins that reside in different g-cells, we must account for the synergy between global and local routes. An example is the net D, E, F, G in Figure 4. As mentioned previously, it is impractical for congestion evaluation to wait until the completion of global routing in order to consider local routes. To simplify our algorithm, we assume that all global routes connect to the center of a g-cell. In this case, we can include the g-cell center as a dummy pin when constructing the Steiner tree to model the local routes. For example, the Steiner tree connecting b, E, F, G is used to calculate the blocked global routing tracks on the four boundaries of g-cell with center b. Similarly, the Steiner tree connecting a, D is used to calculate the blocked tracks corresponding to g-cell with center a. Note that this may cause over-estimation in some cases when connection from D to E, F, G takes fewer tracks than that from a to E, F, G, e.g., D is at the bottom-left corner of the g-cell. To account for this factor, we introduce a parameter p to scale the estimated local resources, where p will be tuned empirically for each technology.

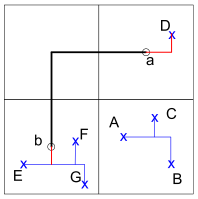


Figure 4: Local routing resources consumed by two nets.

In summary, to consider the effects of local routes in global routing, we add a pre-processing (or pre-routing) step. Specifically, we traverse all the nets, identify the pins inside each g-cell, estimate the local routes using the method presented in this section, and block the global routing tracks from the related g-edges. Local wires inside a g-cell are usually short and for pin accessibility they are typically routed in the second (M2) and third (M3) metal layers during detail routing. Hence, we only block the global routing tracks on g-edges in the M2 and M3 layers during congestion evaluation.

3.2.2 *Method 2: Estimation of local routes based on pin density*

Based on our experiments, Method 1 is reasonably fast and very effective in modeling congestion due to local routes. However, calculating Steiner trees for all the pins of each net in every g-cell can become a productivity bottleneck when congestion analysis is invoked hundreds of times (for example, during physical synthesis). We now propose an alternative method that is simpler and much faster than Method 1, yet equally effective. This method is based on pin density, and does not involve constructing Steiner trees to estimate the local routes. It is based on the following observations:

- Each pin is associated with a set of local wires connected to it.
- The number of pins in a g-cell is a good indicator of the number of local routes, and is a first-order estimate for routing tracks blocked by local routes within the g-cell.

Based on the above observations, we model the local routes in a g-cell by $(k \cdot n)$, where k is a technology-dependent parameter, and n is the number of pins in the g-cell. For each technology node, we empirically determine k by comparing the congestion statistics from our analyzer to those obtained from a reference industrial router. During our experiments on industry netlists, we observed that for a given technology node, k is usually similar across different designs and floorplans. This justifies the effectiveness of Method 2 while using a single k value that is technology and not design dependent.

A key benefit of using this method is that one can easily tune k for more complicated design rules at a given technology. Some design rules (such as lithography constraints) in advanced technologies, or a specific design library (smaller track), may be complicated to model through global routing. These factors may also result in significant detailed routing runtime. Tuning k to address the impacts of these issues can serve as a better guide to routability optimization during a physical synthesis flow.

As before, we use a pre-processing step to use Method 2 in a global routing based congestion evaluation tool. Specifically, we traverse all the g-cells and nets, and count the number of pins (n), inside each g-cell. Following this, we block kn global routing tracks, due to the local routes in each g-cell, on the four g-edges related to the g-cell. Similar to Method 1, we only block the global routing tracks on g-edges in the M2 and M3 layers.

Although Method 2 is empirical and much simpler than Method 1, it yields surprisingly good results. In our experiments, described in Section 5.1, Method 2 achieves a 3.6 times speedup over Method 1 with comparable accuracy.

4. METRICS FOR DESIGN CONGESTION

4.1 Limitations of current metrics

Total overflow (TOF) and maximal overflow (MOF): Naïve implementations of these metrics treat the overflow in each layer as identical; however, this is inaccurate as each layer has a different capacity. Normalizing the overflow to the layer capacity can overcome this issue, but other problems remain. The TOF metric does not capture the hot spots in the congestion map, i.e., the severity of congestion in the worst regions of the chip. MOF fares only slightly better, capturing only the maximum overflow value among all the g-edges in the routing graph. This presents a fairly incomplete picture of the congested regions in the design. Moreover, as pointed out in [2], overflow metrics fluctuate greatly, depending on design size, number of g-edges, number of routing layers, etc.

$\boldsymbol{ACN(20)}$**,** $\boldsymbol{WCI(100)}$ **and** $\boldsymbol{WCI(90)}$: These metrics fail to differentiate between a net spanning a single congested g-edge and one that spans multiple congested g-edges.

Example: Consider two nets in the GRG: *net-A* traverses g-edges with congestion 0.50, 0.70, 0.80, 0.90 and 1.10, while *net-B* traverses g-edges with congestion 0.60, 0.80, 0.95, 1.05 and 1.10. When calculating $ACN(20)$, $WCI(100)$ and $WCI(90)$, both nets will be counted with the same congestion. However, their routability is different: clearly, *net-B* is harder to route compared to *net-A*, as it traverses more number of g-edges with higher congestion. This fact is not captured by these net congestion based metrics.

Additionally, minor design changes can cause large fluctuations in the $WCI(100)$ and $WCI(90)$ metrics.

Example: Assume a design has a g-edge e, with $c_e = 40$, $b_e = 0$ and $w_e = 39$. Assume, that we reroute a net to pass through this g-edge (say, to improve timing). Then the congestion of e becomes 100%, implying that all 40 nets crossing e now have a congestion of 100%. As a result, $WCI(100)$ will now report 40 additional congested nets, when in reality we only rerouted a single net. A similar example applies to the $WCI(90)$ metric. Such instability renders these metrics unsuitable for guiding routability optimization.

Although, $ACN(20)$ avoids large swings due to minor design changes, it suffers from the limitation of not accurately capturing design congestion (demonstrated in Section 5.5).

In addition, existing metrics improperly model the congestion along macro boundaries [1,2], leading to an artificially high reported congestion. Referring to Figure 5, net N routes to a pin on macro block B. Due to the blockage, the congestion of edge e would be rated as being above 90%, but in practice, we find that such nets are easily routable. Including these g-edges with artificially high congestion when calculating the metric introduces unnecessary noise leading to improper estimation of the routability. Note that we only suggest to exclude the edges along macro boundaries when calculating the metric *after* global routing to evaluate the routability, but the high congestion of these edges should *not* be ignored during the global routing process.

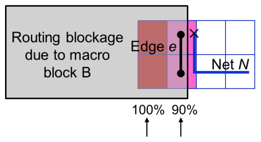


Figure 5: An example showing a net N traversing g-cells that are 90% blocked due to a routing blockage. This leads to artificially high reported congestion for edge e.

4.2 New metric for design congestion

To address the issues with existing metrics, we propose a new metric that is based on the histogram of g-edge congestion.
Our metric has two features:

- It downplays the effects of g-edges with artificially high congestion due to the presence of routing blockages.
- It presents congestion as a histogram, instead of a single number.

To accurately capture the congestion, our metric, denoted as $ACE(x, y)$, computes the average congestion of the top $x\%$ congested g-edges, while ignoring g-edges that are $\geq y\%$ blocked. The role of the parameter y is to void counting the effects of g-edges with artificially high congestion. A typical value for y is 50, implying that all g-edges with $\geq 50\%$ routing blockage are ignored when computing the metric. For convenience, we use $ACE(x)$ to denote $ACE(x, 50)$ in this paper.

In practice, the new metric is most useful when expressed as a vector, for different values of x, e.g., for $x \in \{0.5, 1, 2, 5, 10, 20\}$. $ACE(x)$, for a small value of x, (e.g., $0.5, 1$), provides a highly local view, representing congestion in the regions with the highest contention for wiring resources (hot spots). For larger values of x, (e.g., $10, 20$), it gives a broader picture of the design congestion.

5. VALIDATION AND ANALYSIS

Our proposed techniques, hereafter GLARE, are implemented within a congestion analyzer[2] that performs global routing in the spirit of MaizeRouter [8]. This section provides a detailed analysis of GLARE on advanced industrial designs listed in Table 1. Columns two and three in Table 1 give the design information and columns four and five list the technology specific pin blockage multiplier (k) and Steiner method parameter (p), respectively. These parameters are empirically tuned for each technology. Note that both k and p increase almost twice from 65nm and 45nm to 32nm. This is probably because the design rules in 32nm node become much more complex than previous technologies, and local routing takes more resources than before. All experiments were run on a 64-bit Linux server with 32 CPUs (Xeon® X7560 2.27GHz) and the common color map used for all the congestion plots in this paper is shown in Figure 6.

Table 1: Benchmark designs and the parameters used in the two methods for local routing modeling.

Designs	#Nets	Technology node	Pin blockage multiplier (k)	Steiner method parameter (p)
ckt_12	1660259	65nm	0.050	0.453
ckt_18	528500	65nm	0.050	0.453
ckt_i	350749	45nm	0.050	0.470
ckt_y	321939	45nm	0.050	0.470
ckt_s	1006029	45nm	0.050	0.470
ckt_fb	318116	32nm	0.114	1.000

Sections 5.1 through 5.4 show the impact of modeling local routing resources on routability evaluation and Section 5.5 shows the

[2]Our proposed techniques can also be used in global routers.

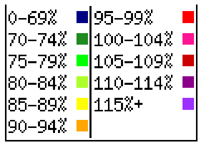


Figure 6: Common color map used for all the congestion plots in this paper.

impact of the proposed congestion metric. For the analyses presented in Sections 5.2 through 5.4, we use the following engines to evaluate the impact of our proposed techniques:

- **Reference Analyzer:** A full-blown industrial router that has a mode for performing congestion analysis with complex modeling of local wires. The reference analyzer is used to judge the quality of all results, and typically runs about 10 times slower than the GLARE based congestion analyzer.
- **Analyzer A**: A fast congestion analyzer that is based on MaizeRouter [8], with the ability to perform global routing on millions of nets in less than 10 minutes.
- **GLARE**: Modification of Analyzer A, incorporating the local routing resources modeling of GLARE.

5.1 Steiner wirelength vs. pin density based modeling of local routing resources

This section compares the runtime and accuracy of the two methods to model local routing resources: Steiner tree wirelength based modeling (Method 1) and pin density based modeling (Method 2).

First, Table 2 compares the runtime of the two methods. In Table 2, "Pre-routing" gives the CPU time for modeling local routing resources, "Total" the total CPU time for congestion analysis, and "Average" the average runtime normalized to Method 2. From Table 2, Method 2 is 3.6 times faster than Method 1 when estimating local routing resources.

Table 2: Runtime comparison of Method 1 and Method 2.

Designs	Method 1 (sec)		Method 2 (sec)	
	Pre-routing	Total	Pre-routing	Total
ckt_12	28.03	401.99	9.00	384.59
ckt_18	8.72	139.50	2.17	133.12
ckt_i	5.24	46.20	1.45	39.73
ckt_y	5.64	48.63	1.93	41.65
ckt_fb	5.12	44.42	1.15	41.36
Average	3.62	1.10	1.00	1.00

Next, Figure 7 shows linear fitting results between local routing resources estimation using Method 1 and Method 2 for design ckt_fb on layer M2. The case for layer M3 is similar and omitted here. Figure 7 shows that with the right values of k and p, we can obtain very good correlation between the two methods.

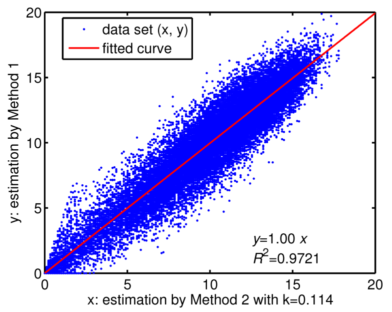


Figure 7: Linear fitting results between local routing resources estimation using Method 1 and Method 2 for design ckt_fb on layer M2.

Based on the results above and in light of its simplicity, we use Method 2 within GLARE to model and account for local routing resources for all subsequent analyses.

5.2 Improving congestion analysis accuracy

This section presents the impact of modeling local routing resources on the overall accuracy of congestion analysis.

Figure 8 shows the results of running the three analyzers on the same placement instance for design ckt_12. From Figure 8(b) we see that using a congestion analyzer with no modeling of local routing resources significantly underestimates the actual congestion. Alternatively, the plot from the GLARE based congestion analyzer (Figure 8(c)) is much closer to the one obtained from the reference analyzer[3], both in terms of the congested regions and their intensity. This result assumes significance in the context of using analyzers within congestion mitigation tools like CRISP [11], where the effectiveness of the tool is highly dependent on accurately identifying the regions of high congestion as well as their relative intensity.

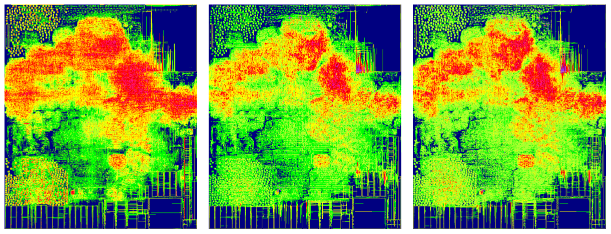


Figure 8: Congestion plots for ckt_12 using a g-cell size of 20 tracks (for GLARE, we set $k = 0.05$).

5.3 Better prediction of detailed routing issues

Often a design that seems routable after global routing can end up with multiple opens/shorts at the end of detailed routing. Early prediction of such issues without performing the time consuming step of detailed routing is highly beneficial as it enables designers to take appropriate measures, thereby improving overall turn-around time for design closure. This section demonstrates that our proposed techniques are able to predict detailed routing opens/shorts with higher fidelity as compared to existing methods.

As an example, we next compare the opens/shorts plots with the congestion plots from Analyzer A and GLARE for circuit ck_fb. Figure 9 shows a plot of the opens/shorts for design ckt_fb during an intermediate stage of an industrial strength detailed router. These opens/shorts indicate the problematic locations in detailed routing, which, in our experience, are usually due to high local congestion at these locations. Figure 10 shows the congestion plots generated by Analyzer A (Figure 10(a)) and the GLARE based analyzer (Figure 10(b)). Comparing Figure 9 with Figure 10(b), we see that the GLARE based analyzer clearly indicates congested hot spots, which translate to the problematic regions for detailed routing – a fact not captured by Analyzer A.

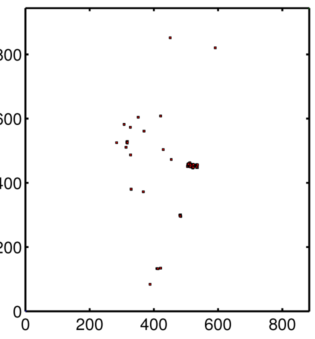


Figure 9: Opens and shorts for ckt_fb during detailed routing.

To quantitatively measure the predictability of the analyzers, we determine the ratio of the number of opens/shorts present in g-cells with global congestion greater than 85% to the total number of opens/shorts in the design. We call this as *match ratio*, and by

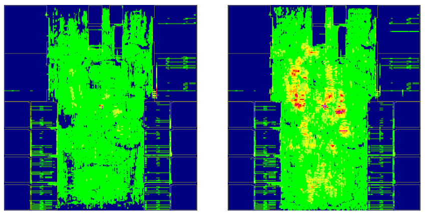


Figure 10: Congestion plots for ckt_fb. The GLARE based congestion analyzer predicts the problematic regions for detailed routing with higher fidelity (for GLARE, we set $k = 0.114$).

definition it measures the percentage of opens/shorts in the most congested regions of the design. We use a threshold of 85% based on prior experience that regions with high global congestion are usually problematic for detailed routing. Using this method, the *match ratios* for Analyzer A and the GLARE based analyzer are 0.20 and 0.87, respectively. Coupled with the congestion plot in Figure 10(b), this further demonstrates the effectiveness of the GLARE based congestion analyzer in predicting detailed routing opens/shorts.

5.4 Accelerating congestion analysis by using a larger g-cell

Since our method can accurately incorporate the effects of local wiring within a g-cell, it provides the freedom to increase the size of the g-cell, thereby accelerating congestion analysis. We demonstrate this by way of running the different analyzers with varying g-cell sizes as outlined below:

- Reference.20: Reference Analyzer, g-cell size = 20 tracks.
- Analyzer A.20: Analyzer A, g-cell size = 20 tracks.
- Analyzer A.80: Analyzer A, g-cell size = 80 tracks.
- GLARE.20: GLARE based analyzer, g-cell size = 20 tracks.
- GLARE.80: GLARE based analyzer, g-cell size = 80 tracks.

Figure 11 shows the results of running the different analyzers on identical placements for design ckt_12. As before, the congestion plot from the Reference Analyzer (Figure 11(a)) is considered most accurate, and used to judge the quality of all results. From Figures 11(b) and 11(c), it is apparent that the quality of the congestion analysis using Analyzer A deteriorates significantly with an increase in the g-cell size. Alternatively, Figures 11(d) and 11(e) demonstrate that the GLARE based analyzer with local routing resources modeling is still able to predict the congested hot spots with reasonable accuracy with an increase in the g-cell size. In addition, the runtime is reduced from 620 sec to 248 sec – a 60% speedup.

5.5 Comparison of routability metrics

Visual inspection of a congestion plot is widely used to quickly evaluate the routability of a design point. We now demonstrate that our new metric can capture a congestion plot with higher fidelity compared to prior metrics for routability evaluation. Consider Figure 12 displaying the congestion plots from two global routing solutions on identical placements for the design ckt_s. The corresponding values for the different congestion metrics are given in Table 3. For the new metric, the congestion is expressed as an ordered pair representing (Horizontal, Vertical) layer congestion.

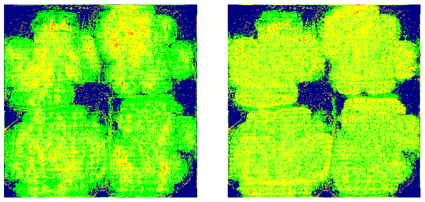
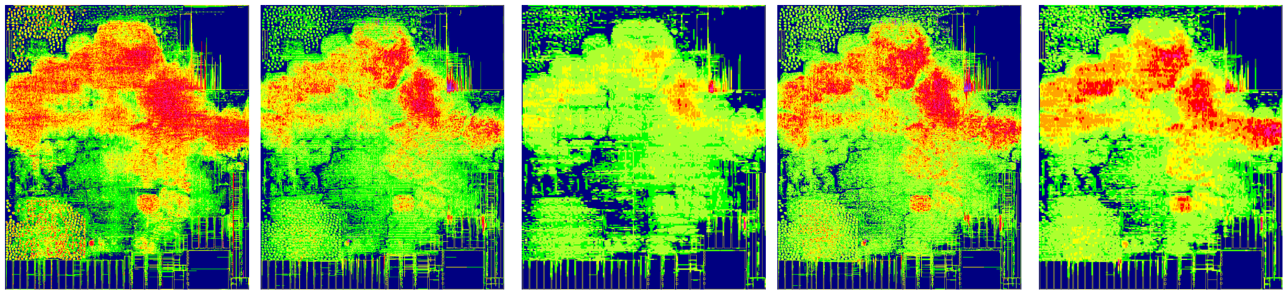


Figure 12: Congestion plots for two routing solutions on design ckt_s.

From Table 3, the overflow-based metrics[4] indicate that Solution 1 has better congestion, while the net congestion based metrics indi-

[3]It is expected that errors remain in the results generated by the GLARE-based congestion analyzer compared to the reference analyzer, since it runs much faster, and does not work as hard as the reference analyzer. However, GLARE-based congestion analyzer can generally predict the hot spots well.

[4]To counteract the drawbacks of overflow metrics discussed earlier, when



(a) Reference.20 (11287 sec) (b) Analyzer A.20 (557 sec) (c) Analyzer A.80 (207 sec) (d) GLARE.20 (620 sec) (e) GLARE.80 (248 sec)

Figure 11: Congestion plots and runtime for different analyzers with varying g-cell sizes for the design ckt_12.

cate that Solution 2 is better, as $ACN(20)$ of Solution 2 is better than that of Solution 1, even though $WCI(90)$ and $WCI(100)$ are worse.

However, a visual examination of the congestion plots indicates that they are quite similar, demonstrating the deficiencies in the existing metrics. Alternatively, our new metric correctly identifies the congestion of these two routing solutions to be similar.

Table 3: Congestion metrics for two routing solutions on design ckt_s.

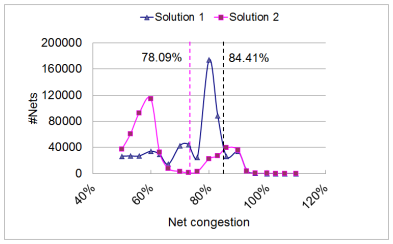
Metrics		Solution 1	Solution 2
Overflow based metrics	TOF	194373	217499
	MOF	10	11
Net congestion based metrics	$ACN(20)$	84.41	77.97
	$WCI(90)$	39494	40548
	$WCI(100)$	274	276
New metric	$ACE(0.5)$	(90.47, 90.54)	(90.27, 90.46)
	$ACE(1)$	(89.23, 89.12)	(89.13, 89.10)
	$ACE(5)$	(85.93, 85.45)	(86.14, 85.49)
	$ACE(10)$	(84.16, 83.39)	(84.59, 83.51)
	$ACE(20)$	(82.48, 81.58)	(82.57, 81.53)

The significant difference in the $ACN(20)$ values for the two comparable routing solutions can be explained by Figure 13(a) which plots the distribution of the worst congestion on the nets. From Figure 13(a), Solution 1 has a considerably higher number of nets in the $[78.09\%, 84.41\%]$ congestion range compared to Solution 2, leading to the difference in the $ACN(20)$ values. In practice, our experience on industry designs is that nets with congestion less than 85% are often easy to route, and considering them within the congestion metric introduces unnecessary noise during routability evaluation. In contrast, looking at Figure 13(b) which plots the distribution of the congestion on the g-edges, we observe the distributions for the two routing solutions to be quite similar above 80%. This explains why the new metric (correctly) rates the two solutions to have similar congestion.

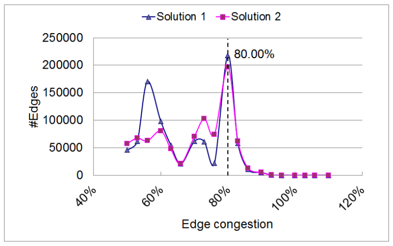
6. CONCLUSION

Fast and accurate routability evaluation techniques are critical to address the increasingly important and difficult issue of routing closure in nanometer-scale physical synthesis. In this work, we have addressed two important aspects of routability evaluation: the accuracy of congestion estimation and a metric for evaluating the routability of a design. We have shown that ignoring the effects of local congestion can result in large errors during congestion analysis. This observation motivates our models for local congestion based on (a) the Steiner tree wirelength of the local routes and (b) the pin density. Experimental results show that the proposed modeling can improve the accuracy and fidelity of congestion analysis, and better predict detailed routing issues such as opens and shorts. It also enables designers to use larger g-cells to accelerate the process of congestion analysis, thereby speeding design closure. Furthermore, we have analyzed the limitations of existing congestion metrics including overflow, etc., and proposed a new metric based on g-edge congestion. We have demonstrated that our new metric can represent a congestion plot with higher fidelity.

calculating overflow, the capacity is scaled down to 80% of the original, and the overflow is in unit of number of minimum-width tracks, e.g., one overflowed track on a 4X layer would be counted as four in the overflow number.



(a) Distribution of net congestion.



(b) Distribution of g-edge congestion.

Figure 13: Distribution of congestion for two routing solutions of ckt_s.

7. REFERENCES

[1] C. Alpert et al. What makes a design difficult to route. In *Proc. ISPD*, pages 7–12, 2010.

[2] C. Alpert and G. Tellez. The importance of routing congestion analysis. *DAC Knowledge Center Online Article*, 2010. `http://www.dac.com/back_end+topics.aspx?article=47&topic=2`.

[3] Y.-J. Chang et al. NTHU-Route 2.0: A fast and stable global router. In *Proc. ICCAD*, pages 338–343, 2008.

[4] H.-Y. Chen et al. High-performance global routing with fast overflow reduction. In *Proc. ASPDAC*, pages 582–587, 2009.

[5] C. Chu and Y.-C. Wong. Flute: Fast lookup table based rectilinear Steiner minimal tree algorithm for VLSI design. *IEEE Trans. on CAD*, 27(1):70–83, 2008.

[6] J. Lou et al. Estimating routing congestion using probabilistic analysis. *IEEE Trans. on CAD*, 21(1):32–41, 2002.

[7] C. Minsik et al. BoxRouter 2.0: Architecture and implementation of a hybrid and robust global router. In *Proc. ICCAD*, pages 503–508, 2007.

[8] M. D. Moffitt. MaizeRouter: Engineering an effective global router. *IEEE Trans. on CAD*, 27(11):2017–2026, 2008.

[9] M. Pan and C. Chu. FastRoute: A step to integrate global routing into placement. In *Proc. ICCAD*, pages 464–471, 2006.

[10] M. Pan and C. Chu. IPR: An integrated placement and routing algorithm. In *Proc. DAC*, pages 59–62, 2007.

[11] J. Roy et al. CRISP: Congestion reduction by iterated spreading during placement. In *Proc. ICCAD*, pages 357–362, 2009.

[12] H. Shojaei et al. Congestion analysis for global routing via integer programming. In *Proc. ICCAD*, pages 256–262, 2011.

[13] J. Westra et al. Probabilistic congestion prediction. In *Proc. ISPD*, pages 204–209, 2004.

[14] T.-H. Wu et al. A parallel integer programming approach to global routing. In *Proc. DAC*, pages 194–199, 2010.

[15] Y. Xu et al. FastRoute 4.0: Global router with efficient via minimization. In *Proc. ASPDAC*, pages 576–581, 2009.