Planning for Local Net Congestion in Global Routing*

Hamid Shojaei, Azadeh Davoodi
Department of
Electrical & Computer Engineering

Jeffrey T. Linderoth
Department of
Industrial & Systems Engineering

University of Wisconsin - Madison WI 53706
{shojaei,adavoodi,linderoth}@wisc.edu

ABSTRACT

Local nets are a major contributing factor to mismatch between the global routing (GR) and detailed routing (DR) stages. A local net has all its terminals inside one global cell (gcell) and is traditionally ignored during global routing. This work offers two contributions in order to estimate and manage the local nets at the GR stage. First, a procedure is given to generate gcells of non-uniform size in order to *reduce* the number of local nets and thus the cumulative error associated with ignoring or approximating them. Second, we *approximate* the resource usage of local nets at the GR stage by introducing a capacity for each gcell in the GR graph. With these two complementary approaches, we offer a mathematical model for the congestion-aware GR problem that captures local congestion with non-uniform gcells along with other complicating factors of modern designs including variable wire sizes, routing blockages, and virtual pins. A practical routing procedure is presented based on the mathematical model that can solve large industry instances. This procedure is integrated with the CGRIP congestion analysis tool. In the experiments, we evaluate our techniques in planning for local nets during GR while accounting for other sources of congestion using the ISPD11 benchmarks.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*layout*

Keywords

congestion, detailed routing, global routing

1. INTRODUCTION

Many factors complicate the routing process for modern designs. Variation in the the wire sizes and spacings of the metal layers, routing blockages, and *virtual* pins in higher metal layers all contribute to make the routing process more difficult. Additionally, modern designs often have high pin density and vias, which further contribute to congestion.

*This research is supported by National Science Foundation under award CCF-0914981.

ISPD'13, March 24–27, 2013, Stateline, Nevada, USA.

To improve a design's routability, one set of techniques focus on *routability-driven placement* procedures that rely on mechanisms for predicting routing congestion. There are many successful efforts in this area, including [7, 8, 9, 10, 11, 14, 16].

Another avenue for improving routability is to adjust the global routing (GR) procedure to account for modern complicating factors. Along this line, CGRIP [13] is a flexible GR tool that can handle arbitrary routing blockages and virtual pins, and varying wire sizes and spacings. However, CGRIP does not account for local congestion effects. Moreover, CGRIP was primarily tuned to provide a rapid estimation of congestion by imposing a small runtime budget—much smaller than typically spent during GR *on unroutable designs* which contain routing congestion.

The focus of this work is to improve routability by modifying the GR procedure to account for congestion caused by local nets, in addition to the complicating factors already handled by CGRIP. A local net has all its terminals in one global cell (gcell). The routing of a local net is determined during the detailed routing stage, and local nets are typically ignored during GR. In many instances, a significant portion of the nets to be routed are local. For example, for the ISPD11 benchmarks [15], using the winning placement solutions, on average 31.20% of the nets are local. The aim of this work is to account for these local nets during GR which translates into significant reduction in the effort right after one iteration of ripup-and-reroute during detailed routing.

This work offers two complementary techniques to manage the impact of local nets during GR. First, we propose to *reduce* the number of local nets by using gcells of non-uniform size. A procedure is presented to transform a given GR instance into one with fewer local nets. The procedure results in an increase in the number of global nets however to have a control on the runtime complexity at the GR stage, it keeps the number of gcells intact. So the effort required at the GR stage may increase but a significantly better solution in terms of the induced overflow is observed, just after one iteration of ripup-and-route at the detailed routing stage. Second, we *approximate* the area required to route the local nets and adjust the areas of the affected gcells before GR.

Using these two complementary techniques to approximate and reduce the local nets, a graph model and a mathematical formulation of GR with non-uniform gcells is presented which includes vertex capacity in addition to the conventional edge capacity. Our model also captures other factors contributing to congestion such as varying wire size and spacing, routing blockages, and virtual pins.

The recent work [17] proposes several methods to approximate the routing usage of local nets inside the gcells, all of which are translated into a reduction in the capacities of related edges in the graph model of GR. In this work, we show using a motivational example and experiments that adding a vertex capacity during GR to reflect this local usage, results in GR solutions which provide a better starting point to perform detailed routing in terms of the induced (detailed routing) overflow, compared to solely reducing the edge capacities. The focus of this work is not on computation of the vertex capacity; any model of the usage of routing resources within a gcell, such as the ones explored in [17], can directly be used as the vertex capacities in our framework. Overall, we show to reflect local congestion during GR, the use of varying vertex capacities together with (unreduced) edge capacities avoid cutting the size of the feasible search space which may otherwise happen if the edge capacities are solely reduced and vertex capacities are not used.

Our experiments are conducted using the recently-released ISPD11 benchmarks [15] which capture a large set of the factors contributing to routing congestion in modern designs. To the best of our knowledge, there is no work in the open literature *on global routing* which have used this challenging set of benchmarks to evaluate routing congestion. In fact we show in an experiment that ignoring these factors during GR and just considering local congestion is not effective in reducing the actual routing congestion.

Our ideas are implemented into a practical routing framework which handles large industry-sized instances. The practical realization is based on integration with the CGRIP congestion analysis tool [13], revising its functionality to suit the goals of this research effort.

2. NON-UNIFORM GCELLS

To reduce the error caused by the modeling approximation of local nets, we propose to define the gcells in a non-uniform manner in order to reduce the number of local nets. Reducing the number of local nets increases the number of global nets (since the total number of nets is constant) and thus creates a more difficult global routing (GR) instance. However, the effort required for detailed routing can be significantly reduced. We will give computational results in Section 5 indicating that the extra effort applied during GR often pays off, generating improved overall designs with less total computing time.

In this section, we first provide mathematical notation for describing local nets in the presence of gcells of unequal size. We then present our non-uniform gcell generation procedure.

2.1 Notation and the Binning Problem

To more accurately account for local effects in GR, we revisit how instances for GR arise out of the design process. At the most fundamental level, we are given a 3-dimensional *placement grid*

$$P = \{0, 1, \ldots X\} \times \{0, 1, \ldots, Y\} \times \{1, \ldots L\}.$$

In most instances, like the ISPD11 benchmark, there are $L = 9$ layers in the grid.

The design problem also consists of a set of nets $\mathcal{N} = \{T_1, T_2, \ldots, T_{|\mathcal{N}|}\}$, where each net $T_n \in \mathcal{N}$ consists of a set of *pin locations* on the placement grid P. As a first step, each net with multiple pins is decomposed into two-terminal subnets based on a minimum spanning tree connecting the pins.

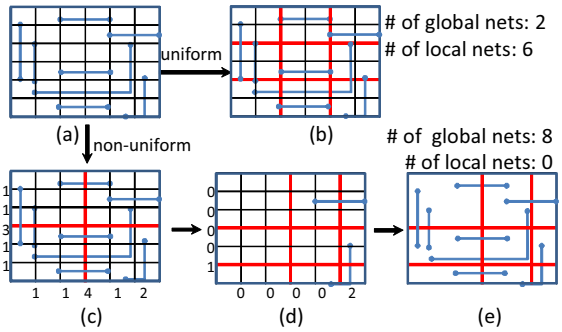

Figure 1: Binning to create non-uniform gcells.

Therefore, after decomposition, each net consists of two *pin locations*: ($T_n = \{p_1^n, p_2^n\}$). The pin locations are coordinates on the placement grid ($p_k^n = (x_k^n, y_k^n, \ell_k^n) \in P$), and typically all pin locations are in the first layer ($\ell_k^n = 1$). However virtual pins outside of the first layer are also possible. For example, in the ISPD11 benchmarks, virtual pins are located at layer 4.

To create an instance that can be solved by GR software, this very detailed placement grid P is replaced with a less-refined version, where coordinates of the placement grid are aggregated into global cells.

Each layer $\ell \in \{1, \ldots, L\}$, consists of a number of *global cells* (gcells),

$$\mathcal{C}_\ell = \{C_{ij\ell}\}_{i=1,\ldots,N_x^\ell, j=1,\ldots,N_y^\ell},$$

where each gcell $C_{ij\ell}$ is a rectangular region on a fixed level of the placement grid P. In our work (and in all benchmark instances), we will assume that a gcell $C_{ij\ell}$ is characterized as a pair of intervals

$$C_{ij\ell} = ([\alpha_i^\ell, \alpha_{i+1}^\ell], [\beta_j^\ell, \beta_{j+1}^\ell]), \quad (1)$$

and the intervals have the property that they partition individual coordinate axes of the placement grid, i.e.

$$0 = \alpha_1^\ell < \alpha_2^\ell < \ldots < \alpha_{N_x+1}^\ell = X \quad (2)$$

$$0 = \beta_1^\ell < \beta_2^\ell < \ldots < \beta_{N_y+1}^\ell = Y \quad (3)$$

We define the selection of the intervals $([\alpha_i^\ell, \alpha_{i+1}^\ell], [\beta_j^\ell, \beta_{j+1}^\ell])$ for the gcells as the *binning problem*. In most GR instances, the gcell intervals have a uniform size. For example in the ISPD11 benchmark instances, all gcell intervals are of length 40. In Section 2.2 we discuss advantages of creating instances whose gcells are of different sizes.

2.2 Binning Procedure

When selecting intervals that define gcells, we do not wish to change the total *number* of gcells, just their individual sizes. Specifically, N_x^ℓ and N_y^ℓ will remain unchanged in Equations (2) and (3), but the cell starting locations $\alpha_i^\ell, \beta_j^\ell$ in Equation (1) will be adjusted by the binning procedure. For purposes of making a 2D-projection of the instance straightforward, the binning procedure will keep all gcells to be of uniform size between layers, i.e. $\alpha_i^{\ell_1} = \alpha_i^{\ell_2}$ and $\beta_j^{\ell_1} = \beta_j^{\ell_2} \quad \forall i, j, \ell_1, \ell_2$.

We explain the procedure for gcell definition using the example shown in Figure 1. We assume that each multi-terminal net is decomposed into two-terminal subnets based on its Minimum Spanning Tree (MST). The MSTs are shown on the placement grid in (a). The standard instance has 9 equal-sized gcells specified by 3x3 grid shown in (b). This gcell configuration results in 2 global nets and 6 local nets.

Our procedure for defining gcells is based on an iterative bi-partitioning. At each iteration, both a horizontal and vertical cut are made to maximize the number of nets that are cut. For example, in (c) the maximum number of nets cut in the horizontal and vertical directions are 3 and 4, respectively, and the corresponding cuts are shown. After each iteration, the nets that are cut are removed from consideration (as shown in (d)). The process completes when the number of intervals N_x^ℓ and N_y^ℓ are of the requisite size. By allowing for gcells of non-uniform size, the number of local nets is reduced to 0, but the number of global nets is increased to 8, as shown in (e).

The binning procedure reduces the number of local nets by maximizing the number of nets that pass a vertical or horizontal cut at each step of the algorithm. It is also computationally useful to consider a parameterized version of this algorithm that controls the tradeoff between the increase in global nets and the decrease in local nets. In the parameterized version of the procedure, when deciding the location of a vertical or horizontal cut at each iteration, we select the cut location that results in the number of cut nets to be closest to ηN_{max}, where N_{max} is the maximum number of nets that can be cut at that iteration. The user-specified parameter η is between 0 and 1 and allows for more fine-tuned control of the eventual number of local nets.

Post-Processing for Local Congestion Balancing: Another important consideration in the binning problem is to generate gcells with "balanced" local congestion. This means to reduce the deviation in local congestion among neighboring gcells. This consideration is helpful in detailed routing because local nets are typically routed inside the corresponding gcell. Reducing this deviation makes it easier to route the local nets during the detailed routing stage. Therefore, after generating non-uniform gcells, we post-process the intervals, perturbing the boundaries of the gcells in order to balance the local congestions among the gcells. For each gcell C_{ijl}, we define the local congestion ratio LC_{ijl} as

$$LC_{ijl} = \frac{R_{ijl}}{A_{ijl}}, \tag{4}$$

where R_{ijl} denotes the routing resources consumed by local nets inside C_{ijl} and A_{ijl} is the area of gcell C_{ijl}. (We discuss a method for estimation of R_{ijl} in the next section.)

The post-processing step is a greedy heuristic with the objective to decrease the deviation of the congestion ratios among the gcells, while ensuring that the number of local and global nets remains the same. The procedure sequentially considers the impact of adjusting each cut line (e.g., up and down for horizontal cuts), computing the updated local congestion ratios and number of global and local nets if the interval boundary was changed. The new cut line location is chosen to be the one that (1) does not change the number of local and global nets; and (2) results in the maximum decrease in total deviation of the gcells' congestion ratios from the average. The latter helps to reduce the deviations in the local congestion ratios among the gcells as a mean to balance the local congestion.

The effect of post-processing is depicted in Figure 2. In Figure 2(a), the congestion ratios are shown for each gcell, calculated assuming the dimensions of each gcell is 4×4, and local congestion is computed using the bounding box of each local net (which is 2 for all local nets). Figures (b) to (d) show the steps of post-processing for the horizontal cuts.

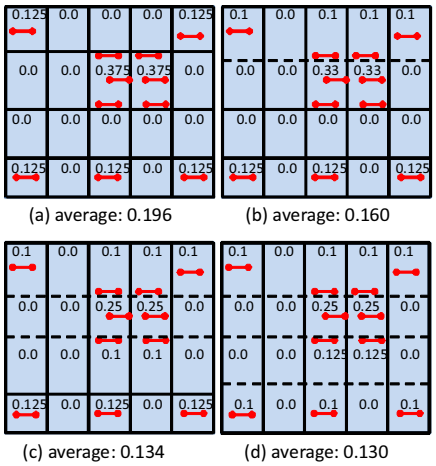

Figure 2: Post-processing example.

In Figure (b), the top cut line (shown by the dashed line) is slightly moved down and as a result some of the local nets are moved from the bottom gcells to their neighboring top gcells. The new congestion ratios corresponding to this step are shown in the gcells and the total deviations in the congestion ratios are reduced from 3.26 to 2.17, compared to the average in each case. Similarly, in Figures 2(c) and (d) the remaining horizontal cut lines are visited and in the end the total deviation in the congestion ratios is reduced from 1.67 to 1.63 in Figures (c) and (d).

An important consideration when applying binning is the amount of routing blockage in the design. For example, in the `sb10` benchmark instance, over 67% of the first four metal layers are defined as routing blockages. In such a case, generating bins of non-uniform size can significantly complicate the global routing procedure and add to its complexity. Therefore, in our framework, we only apply the binning procedure if the amount of routing blockage compared to the total chip area is beyond 50%. In such a case, the less-invasive post-processing step is solely applied.

In Section 5, we show that the above binning strategy is effective for the ISPD11 benchmarks. So far, these benchmarks are the only instances which capture some of the factors that challenge the routability in modern designs and are openly available to the research community. In cases that are not captured by these benchmarks such as designs with more complex routing blockages and/or containing various-sized macros, it would be interesting to investigate our binning strategy when it is selectively applied to "appropriate" regions of the layout.

3. LOCAL CONGESTION MODELING

In this section we describe mathematical models for approximating the routing resources consumed by local nets while considering factors such as non-uniform wire size, wire spacing, and routing blockages. These models are embedded into an Integer Programming (IP) formulation to describe a congestion-aware global routing (GR) problem.

3.1 Local Congestion and Graph Models

To create a GR instance, the gcell information is used to construct a graph $G = (V, E)$, where

$$V = \cup_{\ell \in \mathcal{L}} \mathcal{C}_\ell,$$

with one vertex $v = (i, j, \ell) \in V$ for each gcell $C_{ij\ell}$. Edges $e \in E$ are created between adjacent gcells.

In a given layer, all wiring tracks are oriented in one direction. Additionally there are *vias* to allow for routes to move between adjacent layers. Therefore, each edge $e = ((i_1, j_1, \ell_1), (i_2, j_2, \ell_2)) \in E$ is of one of the following types:

- (EW): if ℓ_1=ℓ_2=ℓ, j_1=j_2, and i_2=i_1+1 or i_2=i_1-1,
- (NS): if ℓ_1=ℓ_2=ℓ, i_1=i_2, and j_2=j_1+1 or j_2=j_1-1,
- (VIA): if $i_1 = i_2$, $j_1 = j_2$ and $\ell_2 = \ell_1$ - 1 or $\ell_2 = \ell_1 + 1$.

We assume without loss of generality if ℓ is odd, edges are of (EW) type, and if ℓ is even, edges are of (NS) type.

Pin locations for each net $T_n \in \mathcal{N}$ are mapped to gcell locations. Specifically, if pin $p_1^n \in C_{ij\ell}$ and $p_2^n \notin C_{ij\ell}$, then vertex $v = (i, j, \ell) \in V$ is a terminal of T_n that must be connected in the GR instance. A net T_n is a *local net* for gcell $C_{ij\ell}$ if $p_1^n, p_2^n \in C_{ij\ell}$. As explained in the introduction, local nets are not explicitly considered by the GR instance, which may result in difficulty at the detailed routing stage.

In the GR instance, (EW) edges and (NS) edges $e = (i_1, j_1, \ell), (i_2, j_2, \ell) \in E$ have a normalized capacity u_e that depends on the length of the interval defining the gcell:

$$u_e = \begin{cases} \frac{\beta^\ell_{j_2} - \beta^\ell_{j_1}}{w^\ell + s^\ell} & \text{if } e \text{ is a EW edge} \\ \frac{\alpha^\ell_{i_2} - \alpha^\ell_{i_1}}{w^\ell + s^\ell} & \text{if } e \text{ is a NS edge} \end{cases} \quad (5)$$

where w^ℓ and s^ℓ denote the wire size and spacing in layer ℓ. By dividing the interval with $w^\ell + s^\ell$, the normalized capacity u_e reflects the number of wiring tracks that can pass between the boundary of adjacent gcells. The first layer $\ell = 1$ is not available for GR in ISPD11 benchmarks, so the capacity is set to 0 for these edges. VIA edges are given a capacity $u_e = \gamma$ that is determined by the technology. For the ISPD11 benchmark instances, $\gamma = \infty$.

To account for local effects, we also consider that each vertex $v = (i, j, \ell) \in V$ has a normalized *vertex capacity* r_v. The vertex capacity is designed to limit the total number of routes that can pass through a vertex (gcell). To capture local effects in a GR instance, the vertex capacity should be reduced based on the area required to route local nets contained in gcell $C_{ij\ell}$. We estimate the area required to route local net $T_n = \{(x_1^n, y_1^n, \ell), (x_2^n, y_2^n, \ell)\}$ in level ℓ by multiplying the wire width by a length equal to the net's half-perimeter bounding box in the lowest possible levels:

$$d_n^\ell = \begin{cases} w^\ell |x_1^n - x_2^n| & \text{if } \ell = 3 \\ w^\ell |y_1^n - y_2^n| & \text{if } \ell = 2. \end{cases}$$

Let $\mathcal{A}_{ij} \subset \mathcal{N}$ be the set of nets local to cell C_{ij1}. The routing resources consumed by local nets at level ℓ is then

$$R_{ij\ell} = \begin{cases} \sum_{n \in \mathcal{A}_{ij}} d_n^\ell & \text{if } \ell = 2, 3 \\ 0 & \text{otherwise.} \end{cases}$$

Note that we make a practical assumption that the local nets of gcell C_{ij1} only impact the capacities corresponding to gcells C_{ij2} and C_{ij3}. *Other models for approximating the routing usage of local nets such as [17] can be incorporated in the above equations.* If there are routing blockages inside a gcell C_{ijl}, the blockage area should be added to $R_{ij\ell}$.

The capacity of vertex $v = (i, j, \ell)$ is the area of the corresponding gcell reduced by the estimated area required for routing local nets. The capacity is normalized by the length of the appropriate gcell interval (depending on whether v is in an (EW) layer or a (NS) layer), so r_v becomes a measure of the number of routes that may cross the gcell. We let

$$r_v = \frac{A_{ij\ell} - R_{ij\ell}}{\lambda_{ij\ell}}, \quad (6)$$

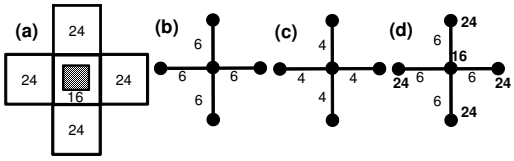

Figure 3: Various global routing graph models.

where $A_{ij\ell} = (\beta_{j+1} - \beta_j)(\alpha_{j+1} - \alpha_j)$ and $\lambda_{ijl} = (\alpha^\ell_{i+1} - \alpha^\ell_i)$ if ℓ is even, and $\lambda_{ijl} = (\beta^\ell_{j+1} - \beta^\ell_j)$ if ℓ is odd.

3.2 Integer Program Model

Input to the integer programming model consists of the grid-graph $G = (V, E)$ and a set of multi-terminal *global* nets denoted by $\mathcal{N} = \{T_1, T_2, \ldots, T_N\}$ with $T_i \subset V$, created as described in Section 3.1. Let $\mathcal{T}_n$ be the *candidate global routes* for net T_n—the collection of all Steiner trees connecting the terminals of T_n. We define the parameters $a_{te} = 1$ if Steiner tree t contains edge $e \in E$, and $a_{te} = 0$ otherwise. Similarly, we define parameters $b_{vt} = 1$ if Steiner tree t contains vertex $v \in V$, and $b_{vt} = 0$ otherwise. The model contains binary decision variables x_t that are equal to 1 if and only if tree $t \in \mathcal{T}_n$ is used to route net T_n. An Integer Program (IP) describing the GR with the normalized vertex capacity is given as follows:

$$\min_{x,o,s} \sum_{n \in \mathcal{N}} \sum_{t \in T_n} c_t x_t + M_1 \sum_{e \in E} o_e + M_2 \sum_{v \in V} s_v \quad \text{(IP-LC)}$$

$$\sum_{t \in \mathcal{T}_n} x_t \geq 1 \qquad \forall n \in \mathcal{N} \quad (\lambda_n)$$

$$\sum_{n \in \mathcal{N}} \sum_{t \in \mathcal{T}_n} a_{et} x_t - o_e \leq u_e \qquad \forall e \in E \quad (\pi_e)$$

$$\sum_{n \in \mathcal{N}} \sum_{t \in \mathcal{T}_n} b_{vt} x_t - s_v \leq r_v \qquad \forall v \in V \quad (\mu_v)$$

$$x_t \in \{0, 1\} \quad \forall n \in \mathcal{N}, \forall t \in \mathcal{T}_n$$

$$o_e, s_v \geq 0 \quad \forall e \in E, \forall v \in V$$

The parameter c_t is the cost of global route t, computed as the *lengths* of its corresponding edges: $c_t = \sum_{e \in E(T)} c_e$. For the non-uniform binning case, the length of each edge is computed as the distance between the centers of the corresponding gcells. In formulation (IP-LC), the first set of inequalities enforces the routing of each net using one of its candidate trees. In the second set of inequalities, o_e is a variable that measures the overflow of the normalized capacity u_e on edge e. In the third set of inequalities, s_v is a variable that measures the overflow of the normalized vertex capacity r_v. The objective is a linear combination of total global routed length, total edge overflow, and total vertex overflow. The two parameters M_1 and M_2 are large numerical constants, so that edge and vertex overflow are minimized.

Motivational Example: Figure 3 depicts a GR instance that demonstrates the utility of vertex capacities. Subfigure (a) shows five gcells with the middle one having high local congestion. The number inside each gcell is an approximation of total number of routes that can cross its four boundaries without causing overflow. A conventional graph model would ignore the local congestion, resulting in the instance depicted in Subfigure (b). In this case, each edge is given a capacity of 6, reflecting the maximum number of routes that can pass the boundary of the two corresponding gcells. Vertex capacities are not included in this model.

Subfigure (c) shows a model that accounts for the local congestion inside the middle gcell by reducing the the number of routes that can pass from each of its boundaries to 4. This case accounts for the local congestion better than (b), but it does so by reducing edge capacities. However, accounting for local congestion by reducing edge capacities may be too restrictive. For example, a solution to the instance depicted in Subfigure (a) can have 6 routes crossing the top boundary, and this solution is excluded by reducing the edge capacities in (c). Subfigure (d) depicts the alternative presented in this work, which uses the same edge capacities as (a). In addition, a vertex capacity of 16 is placed on the middle gcell. This case most accurately models the true instance by accounting for local congestion without unduly restricting the search space.

4. ROUTING FRAMEWORK

In this section we describe our routing framework, which is an extension of CGRIP [13]. By extending from CGRIP, our framework can also account for the same complicating factors— varying wire size and spacing at different metal layers, routing blockages, and virtual pins—as CGRIP. We start by giving a brief overview of CGRIP and then discuss specific extensions to plan for local net congestion.

Overview of CGRIP: The routing procedure in CGRIP starts with a 2D-projection of the instance and the creation of an initial 2D-solution. A rip-up and reroute (RRR) procedure is iteratively applied until a time limit is reached, or no improvement in overflow is possible, similar to other routing procedures [2, 3, 5, 6, 7, 12, 19, 18]. A congestion-aware layer assignment is applied at the last step to account for wire size and spacing, virtual pins, and routing blockages. CGRIP's procedure assumes gcells have uniform size.

CGRIP relies heavily on an IP formulation similar to (IP-LC). A reduced size linear program (RLP), consisting of a subset of the variables of the IP corresponding to critical edges and nets are selected and relaxed to take continuous values. The RLP is solved to determine the amount of utilization on the critical edges which are subsequently fed into the current step of the RRR procedure. The iterative use of these two procedures (i.e., forming an RLP and integrating its solution with the current RRR iteration) leads to a tremendous reduction in total overflow.

Note that CGRIP was designed for rapid congestion analysis. It relies on an input *resolution parameter*. In this extension, we set this parameter to be of maximum resolution. For details about CGRIP, please refer to [13].

Extensions: We now discuss how individual steps in CGRIP are revised to handle the new IP formulation (IP-LC) and the local congestion models with non-uniform gcell sizes.

1) 2D Projection: The 2D projection is done similar to CGRIP. It is done in a straightforward manner using our mathematical notations (presented in Section 3.1) and because even with non-uniform gcell sizes in each layer, we require uniformity across the layers in our gcell generation procedure. In our 2D projection, cells $C_{ij\ell}$, for a fixed location (i, j), whether of uniform or non-uniform size, are aggregated so that there is one gcell at each location (i, j) representing all the layers. Specifically, the 2D graph is $G_{2D} = (V_{2D}, E_{2D})$, where $V_{2D} = \{(i, j)\}_{i=1,\ldots,N_x, j=1,\ldots,N_y}$, and there is an edge $e = ((i_1, j_1), (i_2, j_2)) \in E$ if $i_2 = i_1 + 1$, $i_2 = i_1 - 1$, $j_2 = j_1 + 1$ or $j_2 = j_1 - 1$.

Normalized capacity for edges $e = ((i_1, j_1), (i_2, j_2)) \in E_{2D}$ are computed by summing capacity defined in Eq. (5) over the metal layers:

$$u_e = \sum_{\ell=1}^{\mathcal{L}} u_{(i_1,j_1,\ell),(i_2,j_2,\ell)}.$$

Similarly, vertex capacity for $v = (i, j) \in V_{2D}$ is obtained by summing vertex capacity (Equation 6) over the metal layers:

$$r_v = \sum_{\ell=1}^{\mathcal{L}} r_{i,j,\ell}. \tag{7}$$

2) Initial Solution: To create an initial solution, multi-terminal nets are decomposed into two-terminal subnets according to their Minimum Spanning Tree (MSTs), as in [4, 12, 13]. The same decomposition is used during the procedure to generate non-uniform gcells which was explained in Section 2.

To generate an initial solution, the reduced-sized linear program (RLP) for the modified IP given by formulation (IP-LC) is solved to identify an initial solution from a set of pre-defined candidate routes obtained using maze and pattern routing. This procedure is similar to CGRIP except in the way the RLP is formed which will be explained next.

3) Reduced-Sized Linear Program (RLP): We form a reduced-sized version of (IP-LC) by writing the formulation only for a subset of optimization variables $o_{\hat{e}}$, $s_{\hat{v}}$ and $x_{\hat{t}}$ and the corresponding constraints that include these variables. Variables are identified by first identifying a set of *critical edges*, based on current utilization estimates. (The procedure for identification of critical edges is the same as CGRIP.) For each identified critical edge $\hat{e} = (\hat{v}_1, \hat{v}_2)$, the variable $o_{\hat{e}}$ and two variables $s_{\hat{v}_1}$ and $s_{\hat{v}_2}$ are added. This is because vertex capacities are considered in formulation (IP-LC).

If a critical edge is used in the current solution to route a net T_n, then this net is a *critical net*, and a sufficiently large subset of routes $\hat{t} \in \mathcal{S}_n$ corresponding to candidate routes ($\mathcal{S}_n \subset \mathcal{T}_n$) will have their decision variables $x_{\hat{t}}$ included in the RLP. We note, this process to define the reduced version of formulation (IP-LC) depends on whether gcells are of uniform size. This is because both vertex and edge capacities are computed based on the gcell dimensions. We note however that our non-uniform gcell generation procedure ensures the size of the grid-graph (in term of number edges and vertices) remain the same as the uniform case.

4) Rip-up and Re-Route (RRR): The RRR procedure rips up nets with high overflow and re-routes them. The nets to rip-up are based on the computed overflow in the current solution. The re-route step solves a weighted shortest path problem. In contrast to CGRIP, the shortest path problem in our extension also has weights on vertices. This is because we consider capacity of edges and vertices.

Specifically, in CGRIP, each edge e has a weight of $1+f(\frac{g_e}{u_e})$, where f is an exponential function of an estimate of utilization of edge e, denoted by g_e, and the normalized capacity of e. Similar to CGRIP, if e is a critical edge, then the utilization g_e is the dual value corresponding to the edge capacity constraint for e (π_e shown in (IP-LC)). Otherwise, g_e is computed as the number of global routes that use edge e in the most current routing solution.

In our framework, each edge $e \in E_{2D}$ has an edge weight of $l_e + f(\frac{g_e}{u_e})$, where l_e is the length of edge e, computed as the distance between the centers of the two gcells that edge e connects. The role of l_e in this edge weight expression is to account for the used wirelength associated with each edge.

In contrast, CGRIP uses $l_e = 1$ in its edge weight expression. This is because the cell sizes are equal in CGRIP and the relative contributions of the edges in terms of wirelength are equal to each other.

In our extension, during the re-routing, we also have weights for each vertex $v = (i, j) \in V_{2D}$. The vertex weights are $f(\frac{h_v}{r_v})$, where f is the same function used for the edge weight, h_v is an estimate of the vertex utilization, and r_v is the normalized vertex capacity (7). If v is an endpoint of a critical edge, then its utilization h_v is taken from the dual value corresponding to the vertex capacity constraint for v, denoted by μ_v in formulation (IP-LC). Otherwise, the utilization h_v is taken to be the sum of the edge utilizations for all edges incident to vertex v: $h_v = \sum_{e \in \delta(\{v\})} g_e$.

5) Congestion-Aware Layer Assignment: The RLP and RRR procedures are iterated for a pre-specified time limit, or until no additional overflow improvement is identified. The routes in G_{2D} are then converted into routes on G using a congestion-aware layer assignment procedure. CGRIP uses a greedy procedure for layer assignment which accounts for virtual pins, routing blockages and varying wire sizes and spacing. Here we extend the CGRIP procedure in two ways: (1) we use updated edge capacities which account for local congestion using our model which assumes the local nets are routed at the two lowest layers; and (2) computation of routing resource utilization in the greedy procedure of CGRIP is extended to account for non-uniform gcell dimensions. Specifically, the routing resource of an edge e in G is computed using an estimated length l_e and the corresponding wire size for that layer. The length l_e is computed as the distance between the centers of the two gcells corresponding to the two vertices of edge e.

5. SIMULATION RESULTS

The routing framework described in Section 4 and the binning procedure of Section 2 to create global routing (GR) instances with non-uniform cells were both implemented in C++ and integrated with the CGRIP congestion analysis tool [13]. For the binning procedure, the parameter η was set to 0.9, given significant weight to reducing the number of local nets in the instance. For solving linear programs, CPLEX 12.0 was used. ISPD11 benchmarks were used to validate our framework, and GR instances were created from the winning placement solutions of the ISPD11 contest [1]. For each benchmark, Table 1 shows the placement solution used, the grid size, and the number of nets before and after terminal decomposition. These benchmarks consider non-uniform wire size and spacing, routing blockages, and virtual pins. They are specifically designed to be challenging instances for routability.

GR Variations: We implemented four GR variations:

1. **U-E** (Uniform-Edge): Uniform grid with edge capacity only, without any adjustment for local congestion;
2. **U-AE** (Uniform-Adjusted-Edge): Uniform grid with adjusted edge capacity to capture the impact of local nets without any vertex capacity;
3. **U-AV** (Uniform-Adjusted-Vertex): Uniform grid with unadjusted edge but adjusted vertex capacity to reflect local congestion;
4. **NU-AV** (Nonuniform-Adjusted-Vertex): Non-uniform grid with unadjusted edge capacity and adjusted vertex capacity to reflect local congestion.

Table 1: ISPD 2011 benchmark info

Bench	Placer	Grid Size	#Nets	#2T-Nets
sb1	SimPLR	704x516	822744	2038444
sb2	Ripple	770x1114	990899	2237446
sb4	Ripple	467x415	567607	1316401
sb5	Ripple	774x713	786999	1713307
sb10	RADIANT	638x968	1085737	2579974
sb12	SimPLR	444x518	1293436	3480633
sb15	Ripple	399x495	1080409	2736271
sb18	mPL11	381x404	468918	1395388

The method U-E is identical to CGRIP [13]. Methods U-AV and NU-AV are the ones proposed in this work. They both consider vertex capacity based on local congestion as well as (unadjusted) edge capacity. The only difference between them is whether or not gcells are of uniform size. The method U-AE adjusts the edge capacity u_e to account for local nets similar to Figure 3(c). Specifically, for each edge $e = (i, j)$ of type EW and NS, the edge capacity is reduced to reflect a smaller number of routes that can pass the corresponding boundary of two neighboring gcells. The local congestion is computed using vertex capacities r_i and r_j (given by Equation (6)) for the two end endpoints of an edge. The adjusted edge capacity is then computed by replacing the numerator in Equation (5) by $\min(r_i, r_j)$ indicating that the reduction in the edge capacity is dominated by the gcell with more local congestion. The strategy of reducing the edge capacity to reflect the resource usage due to local congestion is also used in [17].

The above variations are used to create four different GR solutions. The termination criterion in all cases was set to be when no additional improvement in overflow was obtained during the rip-up and reroute (RRR) phase of the algorithm.

Detailed Routing (DR) Emulation: To measure the impact of different GR solutions on the detailed routing stage, we (obviously) require a mechanism for performing detailed routing (DR). In this work, we had to implement our own *detailed routing emulator* to perform this evaluation. We would prefer to use an actual DR tool, but at the time of this writing, there were no DR tools, having an interface that takes a GR solution as input, available to us.

We did obtain a binary of the DR tool `RegularRoute`, kindly shared by the authors of [20]. `RegularRoute` is one of the most recent, competitive academic DR tools. However, at this phase, and similar to other DR tools, `RegularRoute` does not account for the complicating factors introduced in the ISPD11 benchmarks, such as zero metal-1 capacity, virtual pins, and non-uniform wire sizes and spacings. Our attempts to use `RegularRoute` after simplifying the benchmarks (by removing these factors) also failed.

We also carefully evaluated the use of Cadence's `wroute` to perform DR. Unfortunately for our purposes, `wroute`, similar to other commercial detailed routers that we considered, does not have an interface that takes a GR solution as input. Rather, it accepts a *placement instance* as input. This same issue is also mentioned in the paper [20].

Thus, we created our own DR emulator, designed to illustrate the impact of considering local congestion right after the first iteration of detailed routing. We will show that the impact is significantly different among the GR variations that were tested. The goal of our DR emulator is to illustrate the impact of the generated GR solution *immediately after one iteration* in the detailed routing stage, as a relatively-accurate *surrogate measure* to reflect the difficulty of the corresponding detailed routing instance.

Table 2: Comparison of wirelength and total overflow at various stages

	U-E (CGRIP)			U-AE			U-AV			NU-AV		
	GR-OF	**DR-OF**	GR-WL	GR-OF	**DR-OF**	GR-WL	GR-OF	**DR-OF**	GR-WL	GR-OF	**DR-OF**	GR-WL
sb1	0	**23142**	153.36	0	**23020**	154.20	0	**12740**	154.65	0	**806**	154.67
sb2	3168	**18880**	335.80	14496	**18506**	335.36	10526	**13154**	344.46	7780	**9180**	350.31
sb4	228	**28696**	114.40	2024	**27476**	115.12	880	**13296**	119.20	418	**876**	119.88
sb5	0	**10878**	184.74	322	**9256**	187.45	0	**2588**	187.78	450	**1036**	188.82
sb10	124	**84842**	270.32	4502	**73862**	282.65	872	**66780**	281.01	766	**65232**	281.23
sb12	0	**44556**	256.58	274	**44416**	264.11	302	**36414**	259.03	12120	**15732**	261.46
sb15	0	**29982**	191.81	1022	**29800**	192.32	846	**18886**	192.01	2630	**7678**	193.50
sb18	0	**11406**	105.47	0	**11184**	106.90	0	**558**	106.70	0	**444**	107.80
average	1.0X	**1.0X**	1.00X	6.4X	**0.9X**	1.00X	3.8X	**0.7X**	1.00X	6.9X	**0.4X**	1.01X

Our emulator uses a projected DR instance that can be used to estimate the overflow occurring during DR. Specifically, our DR emulator works on a two-layer grid, with one layer containing NS edges and one containing EW edges connected with VIAs. Each route in a given GR solution is then projected to this two-layer grid. Only the original (uniform) gcells are used at this stage to define the gcells where each gcell in the projected instance has a 40x40 detailed routing grid DRG superimposed upon it. This 40x40 granularity gives the same resolution as the placement grid P in the ISPD11 instances, ensuring that pins of all nets (global and local) are vertices of the DR grid. The capacity of each NS/EW edge in the DRG is equal to the number of the NS/EW layers above that edge that do not contain an obstacle at that location.

Global cells are individually routed over the DRG in a sequential, breadth-first manner, starting from bottom left. When visiting gcell c, first the local nets are routed, and then a track assignment is made for all global routes mapped to c in the current GR solution. This process is similar to other published detailed routing algorithms such as [20]. Some track assignments are imposed by neighboring (previously-visited) gcells. For global routes that connect to c through a VIA edge in the GR model, a utilization of one unit of the DRG grid edge inside c is used to reflect this VIA usage. Once the track assignment is made for c, rip-up and reroute (RRR) is applied to route the remaining subnets of each global route and all local nets inside c. In our emulator, we do only one iteration of RRR, so that our emulator shows the immediate impact of the translation of a GR solution into a DR solution. When doing RRR, the same net ordering as in the GR procedure is used. Each net inside c is routed using its shortest path after updating the routing resource usage by the previously-routed nets, similar to the GR framework. Each net, however, is restricted to be routed within a bounding box of its terminals.

Evaluation Metrics: For each generated GR solution, the overflow (denoted by GR-OF) is measured using the (unadjusted) edge capacities that are used in the U-E method. The wirelength of each case (denoted by GR-WL) is also measured. In NU-AV, the wirelength is computed while accounting for non-uniform gcells for fair comparison. For example, an edge in NU-AV which is twice than an edge in U-E due to non-uniform gcells is counted as 2 units of wirelength. For each GR solution, the total overflow computed by the DR emulator (denoted by DR-OF) is computed.

Comparison of Evaluation Metrics: Table 2 shows comparison of GR-OF, DR-OF, and GR-WL for each tool variation. The results confirm the following:

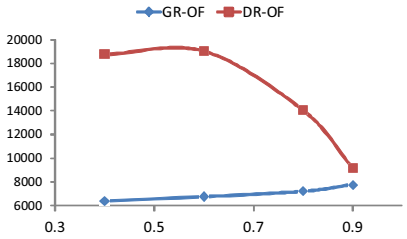

Figure 4: Tradeoff in DR-OF and GR-OF with η.

- Methods U-AE, U-AV, and NU-AV, which account for local nets, result in a reduced DR-OF of 0.9X, 0.7X, and 0.4X, respectively compared to U-E (1.0X).
- Methods U-AE, U-AV, NU-AV all have a significantly higher GR-OF than U-E.
- The GR-WL of methods U-AE, U-AV, NU-AV are up to 1% larger than U-E. Wirelength is increased due to detours as a result of reduced vertex or edge capacities in these methods.

The increase in GR-OF in methods U-AE, U-AV, and NU-AV is to be expected, since solutions are generated for instances with a reduced edge capacity. The resulting solution when evaluated with the original unadjusted edge capacities may have a high overflow compared to U-E. However, in most cases, this increase in the GR-OF is *more than offset* by a decrease in the DR-OF.

Consideration of vertex capacity (U-AV) results in more improvement in DR-OF compared to only reducing the edge capacity (U-AE). It allows reaching higher quality solutions which are excluded from the model of U-AE. Similarly, non-uniform binning results in more improvement in DR-OF.

Impact of Non-Uniform Binning in NU-AV: Our binning procedure is parameterized by a value η that controls the reduction of local nets when generating non-uniform gcells. Figure 4 demonstrate the impact of varying η for the instance `sb2`. (Similar behavior was observed in the other instances.) In the figure, we see that increasing η (which decreases the number of local nets), increases the GR-OF, but decreases the DR-OF. The runtimes have a reverse trade-off: lower η (higher local nets) results in lower GR runtime but higher DR runtime. Note that our binning procedure keeps the total number of gcells the same in all the cases.

When using the binning procedure in NU-AV, we note that for all the benchmarks, DR-OF was improved both with and without the post-processing step. However, post-processing provided additional reduction in these metrics. For example in `sb2` for $\eta = 0.9$, the DR-OF with and without post-processing were 11378 and 9180 respectively —both smaller than DR-OF of the other methods.

Table 3: Comparison of runtime (min) and local nets

Bench	U	NU	U-E		U-AE		U-AV		NU-AV	
	%LC	%LC	GR	**DR**	GR	**DR**	GR	**DR**	GR	**DR**
sb1	30.8	14.1	3	**28**	7	**21**	5	**18**	7	**7**
sb2	28.9	13.5	352	**22**	321	**17**	303	**17**	389	**17**
sb4	35.2	16.8	180	**39**	60	**25**	201	**25**	60	**8**
sb5	29.4	12.2	135	**42**	184	**33**	164	**24**	221	**5**
sb10	34.1	34.0	251	**62**	341	**51**	329	**32**	342	**33**
sb12	28.6	14.6	238	**41**	360	**42**	309	**37**	306	**22**
sb15	34.4	15.8	212	**34**	269	**24**	259	**19**	233	**10**
sb18	28.2	15.0	10	**32**	20	**20**	16	**15**	10	**9**
ave	31.2%	17.0%	1.0X	**1.0X**	1.1X	**0.8X**	1.1X	**0.6X**	1.1X	**0.4X**

Comparison of Runtimes: Table 3 gives a runtime comparison for the different methods. The table shows that the CPU time of the DR emulation is reduced on average by 0.8X, 0.6X, and 0.4X for U-AE, U-AV, NU-AV, respectively, compared to U-E. The GR runtime on average is increased around 10% compared to U-E. Columns 2 and 3 report the percentage of local nets for uniform and non-uniform cases.

The termination condition of CGRIP can be set by the user based on the design flow—whether the user wants a "quick-and-dirty" solution or a solution that spends higher effort to produce a high-quality GR solution. In this work, we set the parameters of CGRIP to reduce the overflow of the GR-solution as much as possible. Specifically, the termination condition for each method is when no improvement in its objective is made for two consecutive RRR iterations. With this termination criterion, the GR runtimes of U-E are 3min and 10min for `sb1` and `sb18`, respectively. However, the runtimes are multiple hours for the other instances. This longer runtime could be reduced at the expense of larger GR-OF values. For example, running CGRIP (the method U-E) for one hour results in GR-OF of 13568 for `sb2`, and overflow exists in 6 of the 8 benchmarks. A comparison of these values to column 2 of Table 2 indicates that the additional effort can significantly reduce GR-OF.

Impact of Local Congestion versus Wire Sizes: Local nets and non-uniform wire sizes are two factors that contribute to congestion, but are ignored or not mentioned in the GR published works. To evaluate the individual impact of these two factors, we conducted a small experiment comparing three GR methods: 1) when local congestion is ignored but non-uniform wire sizes are considered; 2) when non-uniform wire sizes are ignored but local congestion is considered; and 3) when both non-uniform wire size and local congestion are considered. In all cases, the gcells are of uniform size. Case 1 is same as U-E (i.e., CGRIP). Case 3 is U-AV with adjusted vertex capacity. In case 2, we assumed the same wire size and spacing in all layers to be equal to layer 1, which was done by changing the benchmark header line describing the per layer wire size and spacing values. (The routing procedures remain intact.) As a result, if layer 4 had a wire size of 2 units, after getting normalized to wire size of 1 unit in layer 1, then it passes double the number of wires on each edge. This transformation resulted in an increase in the normalized capacity of the edges in the higher layers, since these layers allow for more routing tracks to pass an edge once their wire sizes are reduced to match layer 1. This behavior is the same as the ISPD08 benchmarks. After generating GR and DR solutions, we evaluated each solution using the original wire size using our DR emulator, similar to the previous experiments.

In all cases, the DR-OF, which we are using as a surrogate measure for the goodness of the true DR solution, was significantly reduced by considering both additional complicating factors. For example, for benchmark `sb4`, the DR-OF for cases 1, 2, 3 were 28696, 3348396, and 13296, respectively. Note that case 2 had *significantly* higher DR-OF than the other cases. We conclude that non-uniform wire sizes and local nets are both crucial factors for routability. Our routing models and framework can account for *both* of these factors.

6. CONCLUSIONS

We proposed two techniques for considering local effects during global routing. First, we introduced a technique for constructing GR instances with gcells of non-uniform size that can decrease the number of local nets while controlling the complexity of the GR procedure. Second, we proposed a model to approximate the congestion induced by local nets and incorporated this mathematical model as a vertex capacity constraint into a congestion-aware Integer Programming (IP) formulation. The IP formulation also accounts for non-uniform wire sizes, routing blockages, and virtual pins.

7. REFERENCES

[1] ISPD 2011 routability-driven placement contest [online] http://www.ispd.cc/contests/11/ispd2011_contest.html.

[2] Y.-J. Chang, Y.-T. Lee, T.-C. Wang. NTHU-Route 2.0: A fast and stable global router. In *ICCAD*, pages 338–343, 2008.

[3] Y.-J. Chang, T.-H. Lee, T.-C. Wang GLADE: A modern global router considering layer directives. In *ICCAD*, pages 319–323, 2010.

[4] H.-Y. Chen, C.-H. Hsu, and Y.-W. Chang. High-performance global routing with fast overflow reduction. In *ASP-DAC*, 2009.

[5] M. Cho, K. Lu, K. Yuan, D.Z. Pan. BoxRouter 2.0: A hybrid and robust global router with layer assignment for routability. *TODAES*, 14(2), 2009.

[6] K.-R. Dai, W.-H. Liu, Y.-L. Li. NCTU-GR: Efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing. *TVLSI*, 20(3):459 –472, 2012.

[7] M. Hsu, S. Chou, T.-H. Lin, Y.-W. Chang. Routability-driven analytical placement for mixed-size circuit Designs. In *ICCAD*, pages 80–84, 2011.

[8] X. Hu, T. Huang, L. Xiao, H. Tian, G. Cui, E.F.Y. Young Ripple: an effective routability-driven placer by iterative cell movement. In *ICCAD*, pages 74–79, 2011.

[9] M.-C. Kim, J. Hu, D. Lee, Igor L. Markov A SimPLR method for routability-driven placement. In *ICCAD*, pages 67–73, 2011.

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

[11] J. A. Roy, N. Viswanathan, G.-J. Nam, C.J. Alpert, I.L. Markov CRISP: congestion reduction by iterated spreading during placement. In *ICCAD*, pages 357–362, 2009.

[12] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. *IEEE TCAD*, 27(6):1066–1077, 2008.

[13] H. Shojaei, A. Davoodi, J.T. Linderoth Congestion analysis for global routing via Integer Programming. In *ICCAD*, pages 256–262, 2011.

[14] P. Spindler and F. M. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *DATE*, pages 1226–1231, 2007.

[15] N. Viswanathan, C.J. Alpert, C.C. N. Sze, Z. Li, G.-J. Nam, J.A. Roy. The ISPD-2011 routability-driven placement contest and benchmark suite. In *ISPD*, 2011.

[16] M. Wang, X. Yang, K. Eguro, M. Sarrafzadeh. Multi-center congestion estimation and minimization during placement. In *ISPD*, pages 147–152, 2000.

[17] Y. Wei, C.C.N. Sze, N. Viswanathan, Z. Li, C.J. Alpert, L.N. Reddy, A.D. Huber, G.E. Téllez, D. Keller, S.S. Sapatnekar. Glare: global and local wiring aware routability evaluation. In *DAC*, pages 768–773, 2012.

[18] Y. Xu and C. Chu. MGR: Multi-level global router. In *ICCAD*, pages 250 –255, 2011.

[19] Y. Xu, Y. Zhang, C. Chu. FastRoute 4.0: global router with efficient via minimization. In *ASPDAC*, pages 576–581, 2009.

[20] Y. Zhang and C. Chu. RegularRoute: an efficient detailed router with regular routing patterns. In *ISPD*, 2011.