TraPL: Track Planning of Local Congestion for Global Routing

Daohang Shi, and Azadeh Davoodi
University of Wisconsin - Madison
{dshi7,adavoodi}@wisc.edu

ABSTRACT

We propose a framework to quickly analyze track congestion *inside* each g-cell at the global routing stage. A distinguishing feature of our framework compared to prior work is estimating the locations of vias and partial track utilization by a global segment inside each g-cell for a given global routing solution. We integrate this model with a proposed track assignment algorithm which we show can more effectively reduce track overlaps compared to prior work. A strength of this work is to evaluate the accuracy with respect to an accurate congestion map generated by a commercial detailed router as reference. This work is a step towards bridging the gap between global and detailed routing which is an important obstacle facing modern IC design.

1. INTRODUCTION

Routing has turned into a major bottleneck of the VLSI design flow due to the increasing number of design rules and complexities of advanced fabrication technologies [3, 4, 11, 12, 16]. Many design rules can now impact how routing resources are utilized. Examples include width-length-dependent spacing rules between parallel wires, minimum-area and end-of-line spacing rules [15], and rules to control via size and spacing given the significant increase in metal layer heterogeneity [3, 13].

Within a state-of-the-art VLSI design flow, many design rules are primarily checked and resolved at the detailed routing (DR) stage which has resulted in significant increase in the DR runtime. The starting point of DR is a routing solution provided by the much faster global routing (GR) stage. This is because GR provides an *approximate* routing solution because of using a coarse routing grid composed of g-cells, and routing only a *subset* of the nets in the design known as the global nets (i.e., g-nets). With these approximations, GR can aggressively explore the routing solution space, far beyond what is attainable at DR. However, the utility of GR is significantly diminishing to provide an effective starting point for DR because existing GR techniques have not kept pace with the evolving design rules [3, 13].

DAC '17, June 18-22, 2017, Austin, TX, USA

 ISBN 978-1-4503-4927-7/17/06…5.00
DOI: http://dx.doi.org/10.1145/3061639.3062335

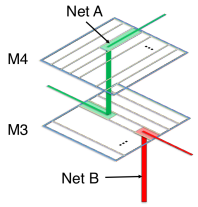

Figure 1: Estimating the via location, thus partial track utilization for the segment of net A on M3 allows assigning the segment of net B to the same track while avoiding overlap.

In this work, we propose a framework to analyze local congestion seen at the DR stage within the GR stage to reduce the gap between GR and DR. Specifically our framework can be used to analyze the routing resource usage seen inside each g-cell. While our framework considers local nets inside the g-cells, its main feature is determining the locations of vias on the routing tracks. Via locations in turn allow estimating partial track utilization to estimate local track congestion (i.e., both utilization and overlap) inside each g-cell. This information can potentially be used as feedback to improve the global router, for example by updating the model of routing resource usage in the global routing grid-graph using vertex or edge capacities [10, 13], defining better overflow metrics such as the via-aware model given in [9], or it can be fed to a routability-driven placer. We note the scope of this work is limited to the analyzer, and not global routing and placement -based optimizations.

There are two ways that our analyzer can be used. **First**, it can analyze an existing track assignment of a GR solution to estimate via locations and partial track utilization inside the g-cells. Recently a number of fast track assignment techniques have been proposed but none of them estimate via locations. We show in our experiments that our analyzer can improve the prediction accuracy of two recent track assignment techniques [14, 17] with respect to the congestion seen at the DR stage obtained by a commercial tool.

Second, our analyzer can be directly used to analyze a GR solution. This is because we also integrated our model of local track congestion with a newly-proposed track assignment algorithm to more effectively reduce track overlaps when each track is fully or partially utilized. For example in Figure (1), estimating via location and partial track utilization of the segment of net A (on layer M3) allows better planning of track assignment for the segment of net B, thus avoiding track overlap inside the g-cell. In this case, our analyzer is also compared with recent track assignment techniques. We show better prediction of track overlaps using the congestion seen at the DR stage as reference.

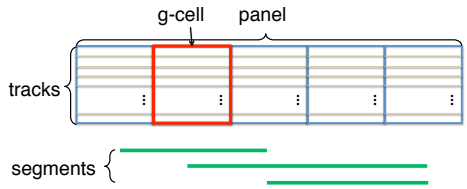

Figure 2: Definition of segments and panel from [17].

Our analyzer runs fast as shown for ISPD 2015 detailed routability-driven placement benchmarks [3].

In the remainder of the paper, we give an overview of related open challenges in Section 2. We explain our model to estimate via locations and partial track assignment in Section 3. Our track assignment algorithm is explained in Section 4. Experimental results are given in Section 5.

2. OPEN CHALLENGES

Here we review related track assignment techniques and identify their common limitations. We only discuss the works which primarily focus on reducing track overlap because it is the objective related to this work. Considering other objectives such as yield, timing, coupling capacitance [6, 7, 8] falls outside the scope of this paper.

First, RegularRoute [17] performs fast detailed routing by doing track assignment on a 'panel-by-panel' basis. As shown in Figure (2), each panel corresponds to a row of g-cells on the same layer. Based on the GR solution, for each panel, a set of segments of the global nets are identified. These segments compete for the routing tracks inside the panel. To determine the track assignment for each segment in a panel, RegularRoute solves a maximum-weight independent set formulation which greedily assigns segments to tracks by prioritizing them according to factors such as segment length, via/pin connection, and density at g-cell boundaries. The assignment aims to maximize the number of non-overlapping segments. RegularRoute then performs additional optimizations to finalize the track assignment. A DR solution may be constructed from the track assignment in the end by using vias to connect consecutive segments of the same net on their assigned tracks. In this case vias may be assumed to be located at the middle of the tracks in the g-cells. The panel-by-panel strategy of RegularRoute is effective in packing as many non-overlapping segments on the tracks of the same panel.

Recently NTA [14] provides an alternative and fast track assignment algorithm. After an initial assignment which includes consideration for local nets, NTA proceeds to a negotiation-based overlap reduction stage. Specifically this stage is done by visiting the segments in a panel-by-panel manner; for each panel, segments with overlap are ripped up and re-assigned to reduce a cost function which depends on metrics such as degree of overlap, wirelength (based on the pin locations), blockage, and history. The rip-up and reroute strategy in NTA allows explicit reduction of degree of overlap between the segments (instead of maximizing number of overlap-free segments [2, 5, 17]).

Both NTA and RegularRoute have the limitation that they consider overlap reduction without accounting for partial track overlap inside the g-cells when a via is used. This is inherently because segments of the same net are viewed independently due to a panel-by-panel processing. However to estimate the via locations, it is essential to view consecutive segments of the same net belonging to different panels.

In this work, we propose a track assignment algorithm which utilizes a net-based processing of segments however the net ordering is guided by a panel-by-panel processing. This hybrid approach allows using the benefits of panel-by-panel assignment but also incorporates a net-by-net processing to estimate the via locations inside the g-cells.

Specifically, we also consider groups of segments on a panel-by-panel basis similar to prior work. However, whenever a segment is considered, then all other segments belonging to the same net (on different panels) will be also be considered and assigned to tracks simultaneously.

Similar to NTA, our work aims to minimize the degree of segment overlap. However we consider both fully and partially -utilized tracks to better estimate and reduce the track overlaps. Also unlike NTA and RR, we actually evaluate the accuracy of our analyzer with the intra-gcell congestion seen at the DR stage and generated by a commercial tool.

3. MODELING THE VIA LOCATIONS

To find the location of a via we assume we are given the track assignments of two consecutive segments which are connected by the via in a global net (i.e., g-net). In case the two segments are apart by more than one layer, we determine the locations of all the vias connecting the intermediate layers which may fall under stacked and unstacked scenarios.

For a given track assignment of a global routing solution, our model can be repeatedly applied to estimate the via location(s) of every pair of segments connected by a via. It can also be applied within our proposed track assignment algorithm which will be explained in Section 4.

We first introduce some notations. Let capacity c_{tg} be the portion of track t falling inside g-cell g. Let utilization u_{tg} be the total length of segment(s) of the g-nets that fall on track t inside g-cell g. Using these quantities, we define a 'total track utilization' seen inside g-cell g as below.

$$\texttt{TrU}_g = \sum_{\forall \text{track } t \text{ in } g} u_{tg} \tag{1}$$

We also define a 'total track overlap' metric for a g-cell g denoted by $\texttt{TrOV}_g$ as below.

$$\texttt{TrOV}_g = \sum_{\forall \text{track } t \text{ in } g} \max(u_{tg} - c_{tg}, 0) \tag{2}$$

The above equation computes the sum of the track overlaps over all tracks in g from the utilization and capacity of each track. Next, we discuss how c_{tg} and u_{tg} are computed in detail in order to compute the $\texttt{TrOV}_g$ and $\texttt{TrU}_g$ metrics.

Track capacity inside a g-cell (c_{tg}): We first initialize c_{tg} to be the length of that boundary of g which runs parallel to the track as shown in Figure (3a). Next, the lengths of pre-routed nets and obstacles on t are deducted from this initial value. In this work, pre-routed nets and obstacles include power/ground mesh, standard cells, macro obstacles, and the detailed routes of the clock nets which are the inputs to our framework. It also includes a detailed routing estimate for each local net which we discuss in Section 4.

Via locations and track utilization inside a g-cell (u_{tg}): Assume we are given a g-cell g that includes some segments of some g-nets and a track t which passes g. The quantity u_{tg} represents how much length is consumed by these segments on track t inside g. (In our track assignment framework, this quantity is initially set to zero and then updated every time a new segment is assigned to t inside g.)

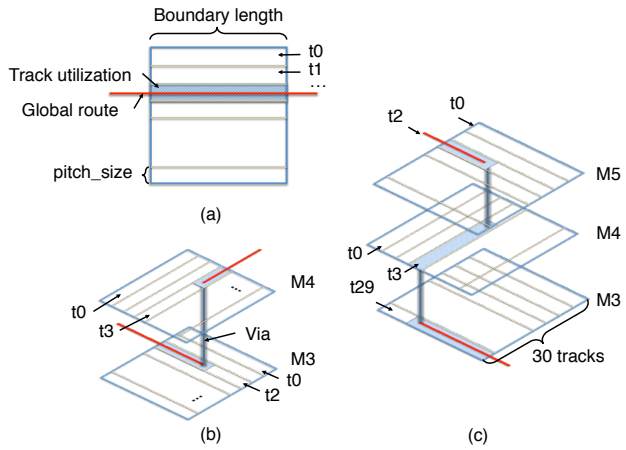

Figure 3: Cases for estimating via locations inside a g-cell

We estimate track utilization for 3 distinct cases. These cases depend on whether a segment connects to another one in an adjacent layer, in which case a via location is found inside the two g-cells that contain the via. The cases also depend on the number of layers that two segments are apart from each other which requires estimating potentially-different locations for each via in the intermediate layers.

For each case, we first discuss how via locations are found. We then assume that track utilization is the corresponding length obtained from the via locations. Note, we assume track utilization includes the area of the vias on the track.

First case, which is the simplest case, is when a segment completely passes through a g-cell. Here no via is used and the track utilization taken by the segment is equal to the length of the boundary of the g-cell as shown in Figure (3a).

Second, if two consecutive segments of a g-net are located on adjacent layers, they are connected by a single via. For a considered track assignment for these two segments, the via location is easily computed from the track indexes (or track locations) of the two segments in the two g-cells. Specifically, the via location lies on the intersection area of the two tracks in 2D space[1].

For example, as shown in Figure (3b), one segment is assigned to track $t2$ on M3 and the other one is assigned to $t3$ on M4. The via location is determined by computing the overlap area of $t2$ and $t3$. Next, the track utilizations are updated using the via locations: for the above example $4 \times$ `pitch_size` units are added to the utilization on $t2$ on M3 and $3 \times$ `pitch_size` units are added to the utilization on track $t3$ on M4 inside the corresponding g-cells.

Third, and the most complex case is when two segments of a g-net are apart by more than one layer. In this case, the two segments need to be connected by multiple vias. Here the vias may be stacked or unstacked.

In the case of unstacked vias which is the more generic case, we consider introducing additional, *local* segments in the intermediate layers. These local segments completely fall inside the g-cells in the intermediate layers. For example, as shown in Figure (3c), the assigned routing tracks of two segments of the g-net are $t29$ on M3 and $t2$ on M5. We introduce a local segment on M4 and estimate a track for it. The local segment on M4 falls completely inside the g-cell and connects the two global segments on M3 and M5.

[1]Note, the analysis relies on the assumption that in practice, the direction of routing tracks alternate in adjacent layers.

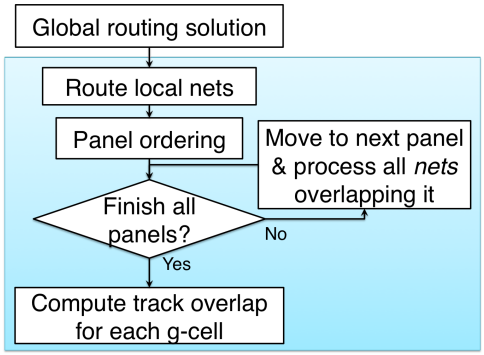

Figure 4: Overview of our framework

To estimate the tracks of the local segments in the intermediate layers, we evaluate all track options for the local segments. Each evaluation considers a temporary track assignment to the local segments. For each track assignment, we compute the via locations; this is done similar to the second case, i.e., by looking at the overlap point of assigned tracks of local and global segments in adjacent layers. Using the via locations, we update the track utilization inside the affected g-cells for each option. In the end, among all track assignment options for the local segment(s) we pick the one which results in minimum $\texttt{TrOV}_g$ metric when added over all the affected g-cells.

In the example in Figure (3c), if track $t3$ is selected on M4, we update the track utilizations as follows: $27 \times$ `pitch_size` units are added to the utilization of track $t29$ on M3, $28 \times$ `pitch_size` units are added to the utilization of track $t3$ on M4, and $4 \times$ `pitch_size` are added to the utilization of track $t2$ on M5. These are determined by looking at the overlapping points of the assigned tracks at the adjacent layers (falling under case 2) to determine the via locations.

Finally, in a special case when the via locations are found to be the same point in different g-cells, the length of the local segment in the intermediate layers is 1. It indicates that the segments of the g-net will be connected by a set of stacked vias to each other.

4. TRACK ASSIGNMENT FRAMEWORK

Figure 4 shows an overview of our framework. We first do fast detailed routing for all the *local* nets (denoted by ℓ-nets). An ℓ-net has all its pins located inside the same g-cell. A detailed route for an ℓ-net is composed of local segments, each of which falls on a specific track inside a g-cell. The local segments are also connected by vias which we determine in our framework. The impact of the detailed routes of the ℓ-nets is reflected in our framework by updating the track utilizations inside the affected g-cells.

To route the local nets, we observe detailed routings of the ℓ-nets are typically on M2 and M3, the lowest two routable layers which allow routing in vertical and horizontal directions. For example, in the detailed routing results generated by Olympus SoC tool [1], 46% of all the detailed wirings of the ℓ-nets are vertical routes in M2 and 44% are horizontal routes in M3, accounting for over 90% of all detailed wirings. With this observation, our framework routes each ℓ-net by only using vertical routes in M2 and horizontal routes in M3 inside the g-cells located above the pins of the ℓ-net.

Specifically, we first create one vertical trunk (V-trunk) on a specific track on M2. The range of the V-trunk covers the span of the ℓ-net based on the locations of its pins.

Algorithm 1

```
1: procedure RocessPanel(panel p)
2:     sort all the global segments on p by length
3:     for each global segment s do
4:         let gnet(s) be the global net containing s
5:         if gnet(s) is NOT 'processed' then
6:             DRoute(gnet(s))
7:             mark gnet(s) 'processed'
8:         end if
9:     end for
10: end procedure
```

After creating the V-trunk, all the pins of an ℓ-net are connected to it by adding horizontal branches (H-branches) on M3. For each ℓ-net, to decide the track to be used as its V-trunk, we pick the track which minimizes the $\texttt{TrOV}_g$ metric. This is done based on the ℓ-nets processed so far and based on other existing static blockages which were given as part of the input to our framework.

After all the ℓ-nets are routed, we sort the panels based on the number of segments of those global nets (g-nets) which fall in each panel. The panels are then visited in descending order. For each visited panel p, we create a detailed route for all the g-nets which have at least one global segment in p. This is assuming the g-net has not been processed before as part of another panel. Algorithm 1 shows the details.

Each time the procedure RoutePanel is called for a panel, it sorts all segments that fall on it by their lengths. The segments are then visited in descending order. For each segment s, the algorithm calls DRouteNet which creates a detailed route for the entire g-net which includes s across multiple panels. This is assuming the net has not been processed/detailed routed before by an already-visited panel.

The detailed route of each g-net is created in order to minimize the sum of the $\texttt{TrOV}_g$ metric over the affected g-cells. (The $\texttt{TrOV}_g$ metric is updated every time a new g-net is processed over the g-cells included in the g-net.) Next we discuss the details of the DRouteNet procedure. Our framework continues until all panels are processed. consecutive tracks **Detailed routing of a single g-net:** The DRouteNet procedure takes as input a g-net (specified by its pin locations and set of global segments), and outputs a detailed route for it. Creating a detailed route includes making a track assignment for each segment. The track assignment should in part ensure the pins of a g-net which fall on specific tracks connect to its segments. It also needs to find the via locations inside the g-cells. The detailed route of a g-net is done with the goal to minimize the sum of $\texttt{TrOV}_g$ over the affected g-cells which in part depends on the track utilizations of the g-nets that have already been detailed routed according to Algorithm 1 and the static obstacles and ℓ-nets that were considered initially by our framework.

In DRouteNet we propose a procedure which simultaneously solves the problem of track assignment and finding via locations. It optimally solves the detailed routing problem for one net, given our track utilization and overlap models.

First, a weighted graph is constructed to represent all track options for each segment of a g-net, and all corresponding vias options for all consecutive segments. Figure (5) shows an example for a two-pin g-net. (Multi-pin nets are decomposed and processed as union of two-pin sub-nets.)

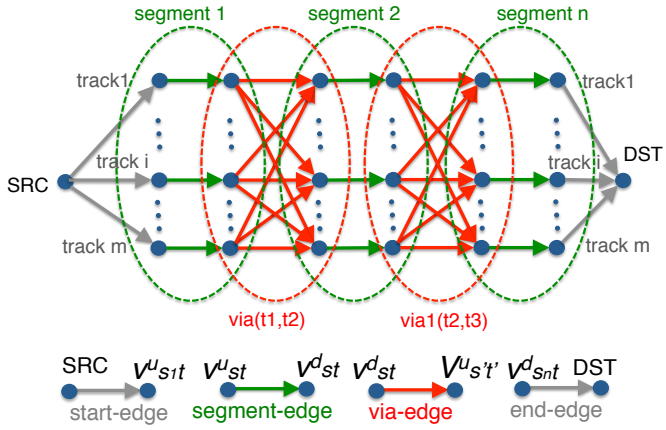

Figure 5: Graph model for detailed routing of one g-net

Two vertices namely SRC and DST which represent the two pins are then created. Next, for every global segment s of a g-net and track option t, we add an upstream and a downstream vertex for the two ends of s. We denote these by v^u_{st} and v^d_{st}, respectively. A directed edge is then added from v^u_{st} to v^d_{st}. Figure (5) shows the set of edges corresponding to the track options for each segment as a group in green color. We refer to this type of edge in the graph as a 'segment-edge'.

Next, for each via connecting two consecutive segments such as s and s', we add directed edges for every possible combination of their track assignments. Figure (5) shows the set of edges corresponding to the via options between two consecutive segments as a group in red color. Specifically, an edge is added from v^d_{st} to $v^u_{s't'}$ as shown in Figure (5). We refer to this type of edge in the graph as a 'via-edge'.

Finally, a set of 'start-edges' are added from the SRC vertex to v^u_{st} if s is the first segment for every feasible track t. Similarly, a set of 'end-edges' are added from v^d_{st} to the DST vertex, if s is the last segment of the global net for every feasible track t connecting to the DST pin. Feasible tracks are those which contain the SRC or DST pins. Figure 5 shows a simplistic example when all tracks are considered feasible.

Next, the edge weights in the graph are computed for each category of edges. First, the weights of start-edges and end-edges are set to 0. For each segment-edge between v^u_{st} and v^d_{st} (assuming s is not the first or the last segment of the g-net) the edge weight is equal to sum of $\texttt{TrOV}_g$ over the affected g-cells. The affected g-cells are the ones that are included in s except the two ending g-cells of a segment. (This is because the track overlaps in the first and last g-cells of a segment will be reflected in the corresponding via-edges.) If s is the first segment of a g-net, the edge weight will additionally include the track overlap of the g-cell containing the SRC pin. Similarly, if s is the last segment, the edge weight will additionally include the track overlap of the g-cell connecting to the DST pin.

Finally, for each via-edge connecting v^d_{st} to $v^u_{s't'}$ (for consecutive segments s and s' on tracks t and t', respectively), the edge weight is defined as follows. We compute the via location and subsequently compute the track overlap in the two g-cells that are connected by the via in s as well as in all the g-cells in the layers between s and s'. This is according to the procedure given in Section 3. The weight of a via-edge is the sum of the track overlaps in these g-cells.

Table I: Comparison of different techniques with respect to Olympus-DR as reference for the 45nm ISPD 2015 benchmarks.

	Error in Track Overlap				Error in Track Utilization					Runtime (seconds)			
	RR	NTA	TraPL	%g-cells	Olympus-GR	RR/NTA	RR+ViA	NTA+ViA	TraPL	RR	NTA	TraPL	Olympus-DR
des_perf_1	19.0%	14.2%	9.4%	98.5	20.7%	21.3%	18.4%	17.7%	11.5%	0.3	3.3	46.1	7200
des_perf_a	8.6%	6.8%	5.5%	46.0	11.4%	11.4%	10.0%	11.3%	8.3%	0.4	11.6	55.9	3000
des_perf_b	7.1%	5.8 %	5.2%	92.0	11.7%	12.0%	10.4%	12.0%	8.1%	0.3	5.2	48.4	1800
edit_dist_a	18.6%	13.6%	10.4%	71.0	15.1%	15.9%	14.0%	14.7%	11.9%	0.8	20.0	115.4	14400
fft_1	15.1%	11.1%	9.0%	99.7	18.4%	19.4%	17.0%	16.1%	12.7%	0.1	1.1	11.6	2400
fft_2	9.6%	6.9%	6.9%	78.9	13.2%	14.5%	13.5%	14.9%	9.3%	0.1	1.8	10.2	1200
fft_a	8.4%	6.8%	7.3%	34.1	12.0%	11.6%	11.1%	10.0%	10.9%	0.2	12.0	16.1	1500
fft_b	7.9%	6.3%	6.9%	31.9	10.1%	9.8%	9.3%	9.3%	10.7%	0.2	4.7	17.7	1200
matrix_mult_1	18.6%	13.8%	9.3%	99.9	19.0%	20.0%	11.2%	17.6%	17.7%	0.5	12.0	65.1	10800
matrix_mult_a	8.0%	6.4%	5.7%	25.0	11.3%	11.1%	9.9%	10.5%	7.9%	0.6	29.3	68.5	1800
matrix_mult_b	5.7%	4.7%	4.8%	30.3	8.7%	8.6%	7.9%	9.5%	7.4%	0.7	24.4	73.6	1800
pci_bridge32_a	5.9%	4.7%	4.7%	69.1	11.2%	9.8%	8.9%	9.9%	7.6%	0.1	1.5	9.1	240
pci_bridge32_b	2.4%	2.1%	2.3%	34.7	6.7%	5.5%	5.2%	6.4%	5.4%	0.1	2.1	9.7	120
Average	**10.4%**	**7.9%**	**6.7%**	**62.4%**	**13.0%**	**13.0%**	**11.8%**	**12.3%**	**9.4%**	**0.33**	**9.92**	**42.11**	**3650**

Once the graph is constructed, we find the min-cost path from `SRC` to `DST`. The path identifies one segment-edge for each segment and one via-edge for each via. It produces the track assignment of all the segments while accurately considering the impact of via locations. The procedure finds the optimal track assignment for the given edge weights. It is also run-time efficient because the graph size only depends on the track count in a g-cell and number of segments in a g-net which are both small in practice.

5. EXPERIMENTAL RESULTS

Evaluation Platform and Benchmarks: We implemented our analyzer in C++ on a Linux machine with a 2.8GHz Intel CPU and 12GB memory. We built an evaluation platform using the Olympus SoC tool of Mentor Graphics [1]. We first read the input files in LEF/DEF format for the 45nm ISPD 2015 detailed routability-driven placement benchmarks [3]. Then we executed the built-in commands of `place_global` and `place_detail` to place each design. Next we ran global and detailed routing for the clock net and then executed the built-in `route_global` for all the signal nets to generate global segments for each one. We refer to this global routing solution as *Olympus-GR* in our experiments. We then proceeded with detailed routing. Specifically we used the `track_route` command which according to the Olympus manual (v2015.2) generates an initial detailed routing solution (while honoring its input global routing solution) and provides information on via location, partial track utilization and track overlap inside the g-cells. This command took longer to run per design as we report. We refer to this solution as *Olympus-DR*.

Moreover, after generating the global routing solution by Olympus-GR, we also fed the global routing solution as input to our analyzer. Specifically, we wrote and sourced a Tcl script in Olympus tool to export the information of built-in DB such as layout dimensions, netlist, placement of cells, exact physical information of various types of blockages and the global routing segments for each global net. The above information was then stored in a custom file and provided as input to the C++ code implementing our analyzer. Our framework computed a total track overlap ($\texttt{TrOV}_g$) and utilization ($\texttt{TrU}_g$) in individual g-cells g which we compare with other approaches and Olympus-GR in terms of 'mismatch' errors computed with respect to Olympus-DR as reference.

Implemented Techniques: We made comparison with our implementation of RegluarRoute [17] (denoted by RR) and with NTA [14] techniques. These techniques both did track routing on the Olympus-GR solution. The RR and NTA techniques were discussed in Section 2.

For fair comparison, both RR and NTA used the same procedure for routing the local nets as the one in our analyzer which we explained in Section 4.

In our experiments, we use our analyzer in two ways. First, we apply our analyzer to the Olympus-GR solution. As explained in Section 4, this variation does track assignment while minimizing track overlaps, accounting for via locations and partial overlaps inside the g-cells. We refer to this variation of our analyzer as *TraPL* (for Track Planning of Local congestion). We compare it to the track assignment solutions of NTA and RR. In the second way, we apply our analyzer to track assignment solutions generated by RR and NTA as input, and estimate the via locations and track utilizations for them. We refer to this variation of our framework as *ViA* (for Via Analyzer)

Comparison of Track Overlap: In our first experiment we compare the $\texttt{TrOV}_g$ metric of different approaches with respect to Olympus-DR as reference. We only considered those g-cells g which contained at least one via; this is because this work is about estimating via locations inside the g-cells. Also the number of such g-cells was high in our benchmarks; the percentage of g-cells containing at least one via compared to total number of g-cells is given in Table I column 5, and was on-average 62.4% across the benchmarks.

Recall the $\texttt{TrOV}_g$ metric reflects the degree of track overlap inside each g-cell and is computed using Equation 2. Since RR and NTA do not explicitly consider via locations, we assumed all vias are located at the middle of the corresponding track inside the g-cell which we used in order to compute the $\texttt{TrOV}_g$ metric. In contract, in TraPL we used our model of via location to compute the track overlap.

Besides computing the $\texttt{TrOV}_g$ metric for each approach, we also compute the 'actual' value of this metric at the DR stage. This is done by measuring this metric for the Olympus-DR solution which provided track assignment and via locations for the same Olympus-GR solution.

Next, we compute the difference in $\texttt{TrOV}_g$ of each approach with Olympus-DR for all g-cells containing a via. We take the average of errors (absolute values) over these g-cells and report it as a percentage of the g-cell capacity. This quantity can be interpreted as a 'mismatch' error in estimating track overlaps inside the g-cells that contain vias.

In Table I we report these values for RR, NTA, and TraPL in columns 2 to 4. TraPL has the smallest error over all approaches, on-average 6.7% across all benchmarks. The maximum improvement in error can be seen in `des_perf_1` benchmark. Here the errors for RR, NTA, and TraPL were 19.0%, 14.2%, and 9.4%, respectively.

Overall NTA and RR have higher error because they do not account for via locations and partial track utilization in overlap estimation during overlap reduction. But our consideration of via locations during overlap reduction results in the smallest mismatch with respect to Olympus-DR.

Comparison of Track Utilization: Table I also shows comparison for total track utilization inside a g-cell denoted by $\texttt{TrU}_g$ and computed by Equation 1. This metric can be very different from track overlap. For example two g-cells may be fully utilized but one may have 0 track overlaps while the other may have a very high overlap because all segments (hypothetically) are assigned to the same track.

For this approach besides comparison with NTA and RR, we additionally compare with the Olympus-GR solution and measure the $\texttt{TrU}_g$ in each case. For the Olympus-GR solution, we do not have (and not need) a track assignment and assume all vias are located at the center of the g-cells. For NTA and RR, we note the value of the $\texttt{TrU}_g$ metric is the same for both so the same column (column 7) in the table represents both techniques. The reason NTA and RR have the same metric per g-cell is because this total track utilization in a g-cell is independent of how track assignment is done. Moreover, since they do not consider via locations, all utilizations are computed with respect to the middle of a track whenever a via is used. Compared to Olympus-GR, the main difference between NTA and RR is consideration of local nets. (Recall we use the exact same local net usage model for TraPL as well. Also please note that local net modeling is not the focus of this paper.)

Additionally, in this experiment we also report results for two cases of NTA+ViA and RR+ViA. In this case, we use our 'via analyzer' to analyze the track assignment solution generated by NTA and RR and estimate the via locations inside the g-cells for each case.

When considering the track utilization metric ($\texttt{TrU}_g$), we similarly measure the absolute value of difference in this metric between each approach and the Olympus-DR. We add this error overall g-cells containing a via and report it as a percentage (normalized to the g-cell capacity) just like the previous experiment. As can be seen our approach (TraPL) drops the average error from about 13% (in Olympus-GR / NTA / RR) to 9.4%.

We can also observe that our analyzer reduces the error in NTA and RR. For example for RR, the error drops from on-average 13% (in RR) to 11.3% (in RR+ViA).

So overall we show in this experiment that TraPL can reduce the mismatch error in estimating the track utilization inside the g-cells. It can also be used as an analyzer to reduce the mismatch of the track assignment solutions generated by RR and NTA by estimating the via locations.

Comparison of Runtime: Table I reports the runtimes of RR, NTA, TraPL, and Olympus-DR. For Olympus-DR our license was limited to single-threaded execution. All techniques ran in a single-threaded mode.

As can be seen, RR, NTA, and TraPL have significantly faster runtimes than Olympus-DR. The runtime of via analyzer ranged from 0.8 to 13 seconds and was on-average 6 seconds. These runtimes are all feasible for an analyzer at the GR stage. The TraPL algorithm supports parallelism by allowing independent groups of panels to be processed simultaneously. We have observed this opportunity in our preliminary analysis of the benchmarks and plan to implement a multi-threaded extension in our future work.

6. CONCLUSIONS

Via locations inside the g-cells are a source of mismatch between GR and DR. We presented an analyzer which predicts the via locations inside the g-cells–both for an existing track assignment solution as well as directly for an existing GR solution. We showed our analyze can reduce the mismatch error between GR and DR (both in computing the track overlap and track utilization) inside the g-cells. Compared to other related work on this topic, our work is the only one that focuses on the mismatch with respect to DR and verifies the prediction accuracy with respect to what is actually seen at the detailed routing stage, and uses a commercial detailed router.

7. ACKNOWLEDGEMENT

This research has been supported by NSF award # 1608040.

8. REFERENCES

[1] Olympus-SoC: Place and route for advanced node designs.

[2] Shabbir H. Batterywala, Narendra V. Shenoy, William Nicholls, and Hai Zhou. Track assignment: a desirable intermediate step between global routing and detailed routing. In *International Conf. on Computer-Aided Design*, 2002.

[3] Ismail S. Bustany, David G. Chinnery, Joseph R. Shinnerl, and Vladimir Yutsis. ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement. In *International Symp. on Physical Design*, 2015.

[4] C. J. Alpert and Z. Li and M. D. Moffitt and G.-J. Nam and J. A. Roy and G. E. Téllez. What makes a design difficult to route. In *International Symp. on Physical Design*, 2010.

[5] Y.N. Chang, Y.L. Li, W.T. Lin, and W.N. Cheng. Non-slicing floorplanning-based crosstalk reduction on gridless track assignment for a gridless routing system with fast pseudo-tile extraction. In *International Symp. on Physical Design*, 2008.

[6] M. Cho, H. Xiang, R. Puri, and D.Z. Pan. Track Routing and Optimization for Yield. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 27(5):872–882, 2008.

[7] X. Gao and L. Macchiarlo. Track routing optimizing timing and yield. In *Asia and South-Pacific Design Automation Conf.*, 2011.

[8] J. Hu, M. Zhao, R.N. Mahapatra, and D. Wu. Timing driven track routing considering coupling capacitance. In *Asia and South-Pacific Design Automation Conf.*, 2005.

[9] D. Shi, E. Tashjian, and A. Davoodi. Dynamic planning of local congestion from varying-size vias for global routing layer assignment. In *Asia and South-Pacific Design Automation Conf.*, 2016.

[10] H. Shojaei, A. Davoodi, and J. T. Linderoth. Planning for local net congestion in global routing. In *International Symp. on Physical Design*, 2013.

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

[12] N. Viswanathan, C. J. Alpert, C. C. N. Sze, Z. Li, and Y. Wei. The DAC 2012 routability-driven placement contest and benchmark suite. In *Design Automation Conf.*, 2012.

[13] Y. Wei, C. C. N. Sze, N. Viswanathan, Z. Li, C. J. Alpert, L. N. Reddy, A. D. Huber, G. E. Téllez, D. Keller, and S. S. Sapatnekar. GLARE: global and local wiring aware routability evaluation. In *Design Automation Conf.*, 2012.

[14] M. Wong, W. Liu, and T.C. Wang. Negotiation-based track assignment considering local nets. In *Asia and South-Pacific Design Automation Conf.*, 2016.

[15] X. Xu, B. Cline, G. Yeric, B. Yu, and D. Z. Pan. Self-aligned double patterning aware pin access and standard cell layout co-optimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 34(5):699–712, 2015.

[16] V. Yutsis, I. S. Bustany, D. Chinnery, J. Shinnerl, and Wen-Hao Liu. ISPD 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement. In *International Symp. on Physical Design*, 2014.

[17] Y. Zhang and C. Chu. RegularRoute: an efficient detailed router with regular routing patterns. In *International Symp. on Physical Design*, 2011.