Intrusive Routing for Improved Standard Cell Pin Access

Vishesh Dokania (vdokania@ucla.edu)
Advisor: Prof. Puneet Gupta (puneet@ee.ucla.edu)
Department of Electrical Engineering, UCLA

***Abstract*—Routability in a standard cell-based physical design flow has emerged as a critical objective in order to meet tight scaling and area requirements while conforming to the increasingly complex design rules. Access to the cell I/O pins can become severely limited based on constraints such as local routing resources, number of tracks, pre-existing nets blocking approach directions, and so on. In this work, we explore a novel approach for enhancing the capabilities of a commercial router aimed at improving pin access and consequently design area utilization. In the proposed intrusive routing methodology, the router would be given the ability to rip-up and change the pre-existing routing configurations inside standard cells, in order to try and achieve a better solution for the particular local scenario. Several approaches are discussed that trade-off solution optimality with runtime and output reliability. We have implemented a framework tool flow enabling one of the different proposed ideas to validate the idea. As a baseline, this proof-of-concept, sub-optimal implementation achieves equivalent utilization compared to a standard methodology for various routing-constrained design benchmarks. Having thus validated our flow, we seek to incrementally extend the framework towards a complete implementation in the future.**

I. Introduction

As device scaling continues into the sub-10nm regime, the problem size/complexity for key physical design steps has gone up considerably. Standard cells are densely packed into rows to achieve the maximum possible area utilization, causing increasingly high pin density/congestion and scarce local routing resources. Out of various steps, design routability satisfying advanced, highly complex process technology rules has emerged as a primary objective of physical design [7], [1]. Imposed DFM constraints such as lithographic considerations, CMP, signal integrity, etc. require continually updated algorithms to effectively automate complex foundry requirements.

Standard cell libraries are usually carefully designed with generous manual intervention in order to achieve optimum cell area, pin placement, compliance with design rules, and conformity with other cells for abutment. Each cell would be replicated a large number of times for implementation of the required design, warranting high effort in optimization of the basic blocks.

These standard cell blocks contain pre-existing internal routing on the lowest metal layers. Each layer has a defined preferred direction (horizontal or vertical) and routing tracks with fixed width and spacing. A subset of these metal polygons are designated as I/O pins for inter-cell routing. These pins can quickly get access-limited for increasingly complex cells. Internal routes would block pin access from one or more directions, causing high local congestion as the router creates jogs around these blockages. Pin access is thus quickly evolving into an important bottleneck for detailed routing [5], [4].

At a high level, this issue would traditionally be solved with approaches along the following lines:

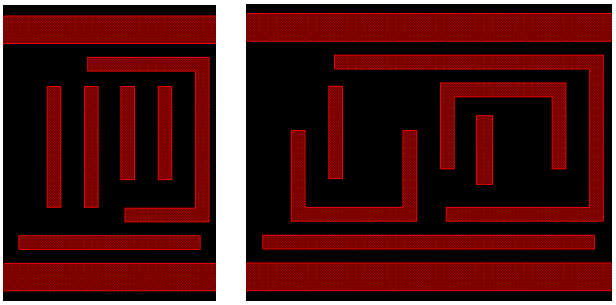
1) Space cells apart: Cells could be packed more loosely, adding white space between them (later occupied by filler cells). However, this goes against our objective of achieving the maximum possible area utilization for given design rules. Moreover, longer wirelengths to connect spaced cells would make it more difficult to achieve performance (critical path delay) and power targets, and cause higher crosstalk/poorer signal integrity [3].

2) Add metal layers: More routing layers could be added so as to enable dropping of via stacks from higher layers to connect to inaccessible pins. This approach would have cost constraints in a commercial setup since each new metal layer would require an additional mask and fabrication procedure in the foundry.

3) Ripup-and-reroute: In a post-routing flow, critical nets could be filtered, removed and rerouted as Engineering Change Order (ECO) routing steps to try and improve the routing solution. However, this has a highly heuristic nature, and involves runtime overhead and high designer effort. Moreover, traditional ripup-and-reroute does not solve the issue if pin blockage is from internal routing.

Even if there exists an optimal initial routing configuration for each cell, it would still be agnostic of the actual design implementation. Pin access considerations would have high dependence on factors such as local routing resources, congestion, and the direction the routed net wants to connect from. Any one given initial standard cell implementation would thus be sub-optimal for a large number of its instances in the design.

In this work, we explore the merits of a possible enhancement to the general methodology of EDA tool flows - allowing the inter-cell design router to intrude inside the library standard cells and change internal routing configurations. This is aimed at leveraging information of actual cell placement, routing resources and local congestion, and utilizing the sophisticated algorithms already employed by commercial routers to possibly find a better routing solution for the full metal layer stack. This should ultimately allow packing cells closer together than the standard flows, and hence improve area utilization.



(a) Cell 1 (b) Cell 2

Fig. 1. Alternate configurations for the same logic cell. Cell 1 has better pin access even though Cell 2 is wider.

The rest of this report is organized as follows. Section II covers background in the form of important considerations that affect pin access. Section III discusses various approaches to implement the proposed intrusive routing scheme. Section IV provides details specific to the chosen implementation and evaluation setup. Finally, Section V presents obtained results, and Section VI talks about various future directions of this work.

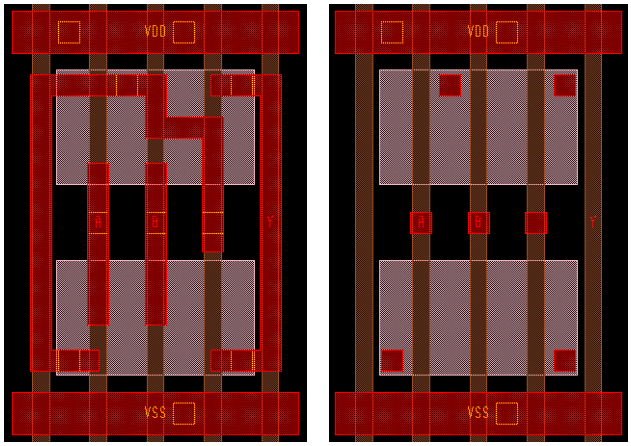
II. Pin Access

A standard cell design implementing a particular logic function and satisfying a specific set of technology-specific design rules can have a variety of layout configurations, typically available in different sizes. Traditionally, larger cell areas allow a greater number of metal tracks to pass over the cell area, hence improving the router's options to access required pins. Consequently, a larger pin metal shape also aligns with a larger number of tracks. With technology scaling, the number of tracks covering the cell has reduced significantly, making the router's task much more challenging. Pin accessibility has proven to be a bottleneck for local routing [4][2].

Pre-routing can adversely affect the routability of other nets. Inter-cell wiring of one net often affects pin accessibility of another, which makes ordering of routed nets critically important [2]. Going down a level of hierarchy, pre-wiring present inside the standard cells to connect various contact shapes in order to implement the intended logic can also block available tracks for other pins. This indicates possibly worse accessibility even with a larger cell area. Two example configurations of the same logic cell from a library are shown in Fig.1. Even though Cell 2 has more inter-pin spacing, multiple metal shapes wrap around smaller pins to block access from different directions. Thus, Cell 1 potentially offers better local routability in terms of pin access.

The standard solution to address some of these issues is the ripup-and-reroute method [6] which heuristically removes a critical inter-cell routed net that is blocking accessibility for others, and then re-routes it to try and find an alternate non-blocking path. Almost all commercial routers adopt this approach to enable post-route optimization and DRC fixes for an improved solution.

In this work, we are exploring an augmentation to the ripup-and-reroute framework to allow it to modify the default routing



(a) Original cell (b) Complete metal routing rip-off

Fig. 2. Example layout view of a 2-input AND library gate: (a) original, (b) after complete cell metal rip-off. Note the internal net inside the cell - the 6-segment polygon containing the left-most vertical bar.

inside standard cells. Unlike the original idea that always occurs post-routing, the proposed idea can be explored both as a pre-routing as well as post-routing step. We expand on this in Section III.

III. Intrusive Routing Approaches

In this section, we propose different approaches that could be used to leverage the original idea of intrusive routing, enabling the router to modify the internal routing configuration of standard cells using one or more of these. We present these as 3 distinct techniques below.

A. Complete intra-cell routing rip-up

In this approach, we propose ripping-up all existing internal metal routing layers inside the standard cells, while maintaining lower silicon and contact layers. This essentially exposes the task of complete routing inside the standard cell to the design router. As a result, apart from the nets obtained from the design netlist, it now has to also connect the correct sets of contacts within the cell to ensure the same logical function as the original library cell.

To enable this, we would need to keep minimum-sized Metal1 (M1) shapes or "landing pads" covering each contact, and communicate internal connectivity information to the router. This introduces 2 kinds of connectivity: (i) Individual isolated pin shapes of I/O pins that need to be connected, and (ii) internal nets inside the standard cell. This also leads to some interesting sub-cases that we will expand upon in Section IV. The idea is illustrated in Fig.2.

On one hand, this approach inflates the problem size for the router by a large extent since it has to now connect all the extra endpoints inside cells that it was not originally aware of. This directly leads to larger runtime. There is also the potential risk of losing important cell logic functionality in case of routing failures for the internal nets; it thus requires stringent LVS checks and possibly stricter design rules for enforcement.

However, we argue that this approach should achieve the best routing solution for the proposed idea, as long as the combined design size is within the router's capabilities. Since the router now sees the complete picture of nets inside and outside the cells, it has the freedom to co-optimize I/O pin shapes and cell internal nets ((i) and (ii) described earlier) using algorithms identical to inter-cell signal nets. We can thus leverage the sophisticated capabilities of a commercial router to make congestion-oriented locally-optimal decisions for an improved design routing result.

B. Congestion-based selective intra-cell routing rip-up

While the previous approach can potentially yield the best results, it would do that at the cost of higher runtime. Each net would have additional terminals to connect, along with new within-cell internal nets. Moreover, net reordering might affect detailed routing and require additional ripup-and-reroute steps post-routing.

An alternative approach would then be to recognize routing areas with high congestion and/or design rule violations, and selectively rip-up internal routing of only the cells in those areas. This would require a global routing congestion estimate as the first step, in order to obtain a per-grid cell (gcell) congestion analysis. Areas with the most congestion would then be heuristically replaced by cells with internal routing ripped-off, followed by detailed routing. Alternatively, global and detailed routing could both be run at the start. The recognized routing violations could then be treated as markers for cell metal rip-up, in order to potentially legalize the violations.

Although this could potentially involve multiple iterations, it should be noted that the problem size would get smaller on every iteration as the number of violations go down, finally reaching some defined convergence. A well-designed heuristic could then optimize this approach to trade-off the routing solution with runtime. This approach can thus be treated akin to the standard ripup-and-reroute method, augmented with the ability to affect intra-cell nets. However, the largely empirical router intrusion would not always find the right solution for the local congestion or violation.

C. Multiple cell pin-access configuration swap

As a third alternative offering the other extreme of pros and cons, we could have multiple preset variants of each cell in the library. Unlike the previous approaches, the intra-cell routing would not be ripped off, making sure every preset cell configuration is legal. These presets would be created such that they are each beneficial for different pin access directions. Similar to the previous approach, we could then permit heuristic swapping of these cells by the router based on the true direction of approach, congestion and violations.

This approach has lower risk of routing failures since each pre-created cell configuration can be ensured to be a legal implementation with no future intrusion from the router. The problem size for the design router would also not be inflated since we would not be adding any new nets or terminals. However, this method would be highly dependent on designer expertise and intuition of true pin access possibilities. It would also be heuristic-dependent since the router would need to choose the right cell to swap with.

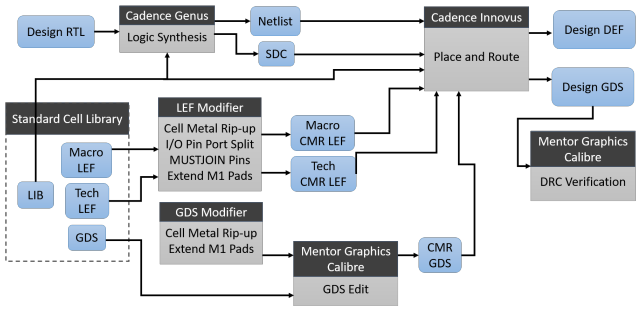


Fig. 3. Intrusive Routing Tool Flow

IV. Methodology and Implementation

In the previous section, we discussed three different approaches for intrusive routing. We now move on to describing the actual methodology followed for their implementation.

The approach for this work has largely been aimed at investigating and defining a concrete tool flow to enable the proposed enhancements to the standard physical design flow. Modifications to the cell library are not traditionally a part of the capabilities of commercial tools. Thus most of the effort was spent towards investigating methods to properly communicate the required library modifications to the commercial router, and thus indirectly enhancing its capabilities.

As a proof-of-concept implementation, the setup has been successfully tested for a basic version of the proposed approach in Section III - complete cell metal rip-up (CMR). The framework can thus be incrementally built upon in order to implement optimized versions of the different proposed approaches in the future.

A. High-level Flow

The overall flow is depicted in Fig.3. We use a predictive 7nm standard cell library for design space exploration. The library contains around 80 standard cells stored primarily in Library Exchange Format (LEF), Liberty model (LIB), and layout GDS. There are separate LEFs containing macro (standard cell) information and technology information.

The library LEFs and GDS are edited by scripts that directly edit their values according to required specifications. This effectively achieves cell metal rip-up by (i) modifying polygon coordinates stores inside the macro LEF, and (ii) modifying appropriate polygon shapes in GDS. All editor scripts have been developed from scratch on Python and Tool Command Language (Tcl).

The input design RTL is synthesized using the Cadence Genus Synthesis Suite, which generates an output gate-level netlist and timing constraints file (SDC). All modified interfacing files are fed to the Cadence Innovus Implementation System, a state-of-the-art commercial physical design suite. Finally, the output GDS from Innovus is checked for DRC violations in Mentor Graphics Calibre.

B. Library Modifications

The most important part of the tool flow enabling standard cell metal rip-up is the Macro LEF. This contains a text-based physical view of the standard cell, defining its size, pins and obstructions. Specifically, each pin definition in the LEF is associated with layer-wise metal polygon shapes represented as coordinators. Thus, editing these polygons directly serves to achieve the required modifications in the standard cells. The same approach can be extended to GDS layouts when viewed in Calibre.

However, simple rip-up of all pre-existing metal shapes is not enough. The major task now is to communicate the connectivity of the remaining shapes inside the cell that are isolated in 2D space. This is achieved by utilizing special property definitions in the LEF standard. Specifics for the 2 types of intra-cell connectivity are described below:

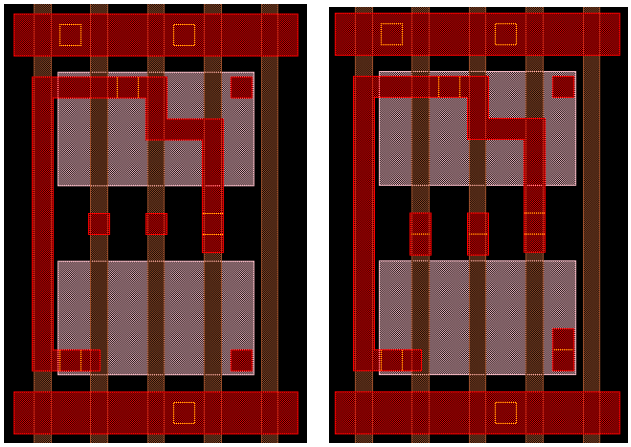
1) Cell I/O signal shapes: These are the interface pins connecting to inter-cell nets. These pins would originally be a single shape in the standard cell definition, which is now ripped-off into isolated shapes covering the contacts. Thus the original net connectivity is broken, and now these additional extra endpoints must be added to the same net. Connectivity is achieved by leveraging a special LEF property called "MUSTJOINALLPORTS". This dictates the router to connect the incoming net to all ports associated with a pin instead of a single shape. Thus if each isolated pin shape described above is treated as a port, this proves to be an effective solution.

2) Cell internal nets: These are nets completely internal to the standard cell (Fig.2) and do not normally show up to the router. Since we now rip them up, new nets must be added in their place to enable the standard cell to perform its designated logic function. Different methodologies can be adopted for this:

- *MUSTJOIN Pins*: These are special pin types in the LEF standard used to specify pre-wiring as guides for nets. The same functionality can be extended to represent completely internal nets. The router then generates implicit nets for these pin types.
- *Add nets in DEF*: As an alternate approach, new nets could be added to the output DEF generated by Innovus at the intermediate stage of the P&R flow. The new nets would be determined based on internal contact connectivity and absolute positions of these isolated pins in the placed design area.
- *Virtual Pins*: LEF standards also define special virtual pins that can be added to existing nets as a new coordinate endpoint. Although this approach should work in theory, it could not be explored because Innovus does not support this syntax.

C. Experimental Setup

To verify functionality of the tool flow, we implemented a basic version of the first approach in Section III as a proof-of-concept. Specifically, we explored the rip-up of all metal polygons inside the cell *except* the internal nets, as shown in Fig.4(a). This partial rip-up of polygons (p-CMR) was then tested across a few benchmark designs. Design area utilization was swept in a heuristic flow to determine the maximum possible utilization that the flow can achieve. The p-CMR utilization was then compared against maximum utilization obtained from a standard flow.



(a) p-CMR (b) p-CMR with M1 extensions

Fig. 4. Layout depicting p-CMR with partially ripped-off cell shapes. The retained internal net is clearly seen in (a). M1 landing pad extension is demonstrated in (b). Note the unextended shape on the top-right corner.

TABLE I
EVALUATED BENCHMARK DESIGNS

Design	Size (#Gates)	Top Routing Layer
cortexm0ds	9k	M4
aes_cipher_top	14k	M4
fpu	33k	M3

A key observation we made during our test runs was that of a new routability constraint that is introduced by our CMR methodology - metal track misalignment. Comparing Fig.2(a) with Fig.4(a), it can be easily observed that the longer vertical segments of original I/O pin shapes would be aligned with multiple horizontal tracks. However, if we rip-up all the M1 in the cell so as to only cover the contact shapes, there is now a misalignment between the M1 and M2 tracks. Even if the horizontal M2 track covers one such M1 shape, the contacts are unevenly spaced in the vertical direction such that it is impossible to cover all of them while keeping a constant track pitch. To resolve this issue, we extend the default M1 landing pads to additionally cover the width of atleast horizontal track, as shown in Fig.4(b).

We chose a basic approach for pad extensions in our proof-of-concept implementation - extend all pads in one direction, then remove any extensions that conflict with existing internal shapes. Thus the shape in the top-right corner of Fig.4(b) is not granted an extension. We note that this naive approach certainly brings in sub-optimalities, but allows us to quickly evaluate our implementation. Further extensions to this approach are discussed in Section VI.

Utilization sweep was carried out for a confidence interval of $\pm 0.1\%$. A cursory value of 50 violations was chosen as DRC threshold to evaluate "legal" routing output. The

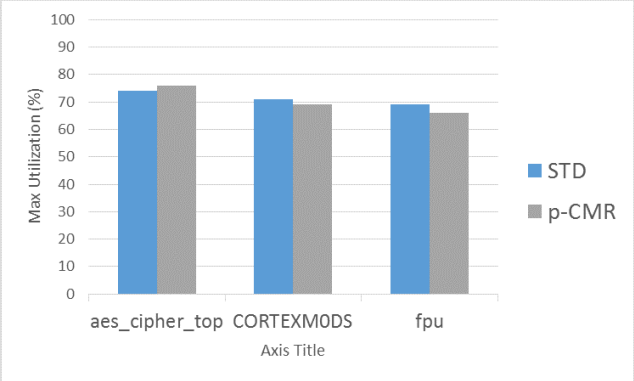


Fig. 5. Maximum Utilization Achieved - Standard vs p-CMR

highest permissible routing layer is concurrently pushed down to evaluate a fully constrained version of the flow. Final results are reported for the lowest legal top routing layer that warranted a fair comparison.

Evaluated benchmark designs are listed in Table I along with design size and top routing layer chosen. All designs were obtained from the IWLS 2005 benchmark suite, containing samples compiled from various sources such as OpenCores. The particular benchmarks were chosen because they were found to be the only routing constrained designs among around 8 tested designs from the same suite. Only a constrained design would give meaningful results based on maximum utilization achieved.

V. Results

Based on the implemented intrusive routing framework discussed in Section IV, we perform a comparative analysis for the partial CMR scheme (p-CMR) against the default flow. The 3 benchmarks designs mentioned in Table I were found to be routing constrained, reaching a similar maximum utilization. The obtained results are shown in Fig.5.

As observed, p-CMR performs equivalently to the standard flow. It obtains a 2% utilization improvement for aes_cipher_top, while performing 2-3% worse than the standard flow for cortexm0ds and fpu. Given the heuristic nature of commercial CAD tools and propensity to converge at local optima, numbers this close should not be assigned too much meaning. Essentially, p-CMR is shown to perform equivalently to the standard flow.

It should be noted that the results presented in this report are a preliminary check to validate the proposed ideas and developed methodology. As discussed, we only implement the most basic version of the proposed intrusive routing approaches in order to establish baseline performance. Even though the router is given access to shapes inside the cell, it is limited by: (a) pre-existing internal nets that are not ripped-up and can thus offer pin access blockages, (b) partial M1 landing pad extensions, since any extensions that violate spacing against existing internal nets are omitted. Both of these considerations would be directly alleviated with a full CMR implementation.

From this point forth, there should be a lot of room for incremental improvements to the implementation seeking to achieve meaningful utilization improvements. Various future directions are discussed in the next section.

VI. Future Work

After the validation of basic methodology presented in the previous section, there are a lot of different approaches worth exploring to evaluate the merits of the proposed idea. A few of these are summarized below:

- *Full CMR based on MUSTJOIN pins*: All internal nets can be ripped off and added as MUSTJOIN pins to add implicit nets for the router. This approach has been tested to work correctly. However, the internal nets are incorrectly recognized as 'floating' violation by the CAD tool, thus affecting the routing output. Workarounds to this issue should be explored to correctly leverage this scheme.
- *Full CMR based on new DEF nets*: The other approach to add internal nets would be to generate new nets based on absolute pin positions on the design area, and add these to the DEF at the pre-routing stage of the flow. However, there would be conformity issues when comparing the design to the original input netlist and the Liberty model since there would be a mismatch in nets and pins associated with each cell. An appropriate fix to solve this issue should be explored.
- *Smart M1 landing pad extensions*: As discussed, the current approach for creating M1 landing pad extensions in p-CMR is a naive approach for basic validation. The extensions are all carried out in the same direction, and any extension coming too close to another existing shape is removed. This can be significantly improved by allowing the shape to extend in a different direction before extending. Furthermore, M1 landing pads should not really conflict in case of full CMR, except a few pathological cases. The routing solution should thus see improvements based on this.
- *Congestion/violation-based CMR*: As discussed in Section III, selective CMR based on congestion estimates or recorded violations can bring down the problem size significantly, but will have a heuristic nature.
- *Cell variant swapping*: As discussed in Section III, multiple legal cell configurations can be swapped based on true pin access considerations without affecting the job/problem size of the router.

VII. Conclusion

In this work, we explored a novel approach towards enhancing the capabilities of a standard commercial inter-cell router aimed at improving pin access. A number of intrusive routing approaches were discussed to allow the router to rip-up existing routing present inside the standard cells, in the interests of rerouting them into a better configuration for the particular local scenario. Specific implementation methodology was discussed, presenting a complete tool flow capable of granting these extensions to the commercial router. As a

preliminary validation routine, a partial version of the proposed cell metal rip-up technique (p-CMR) was implemented and tested through a comparative analysis. Results for several design benchmarks demonstrated equivalent performance to the standard flow in terms of maximum utilization achieved, hence validating the approach. The framework can now be extended in several directions for a complete implementation in order to evaluate the true merits of the proposed idea.

REFERENCES

[1] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "BoxRouter 2.0: architecture and implementation of a hybrid and robust global router", *IEEE/ACM Int. Conf. Comp.-Aided Des. (ICCAD)*, Nov. 2007.

[2] M-K. Hsu et al., "Design and manufacturing process co-optimization in nano-technology", *IEEE/ACM Int. Conf. Comp.-Aided Des. (ICCAD)*, pp. 574-581, Nov. 2014.

[3] X. Qiu and M. Marek-Sadowska, "Can pin access limit the footprint scaling?", *Des. Automation Conf. (DAC)*, Jun. 2012.

[4] X. Xu, B. Yu, J-R. Gao, C-L. Hsu, and D. Z. Pan, "PARR: Pin-Access Planning and Regular Routing for Self-Aligned Double Patterning", *ACM Trans. Des. Automation of Electronic Syst. (TODAES)*, vol. 21, no. 3, Jul. 2016.

[5] T. Taghavi et al., "New placement prediction and mitigation techniques for local routing congestion", *IEEE/ACM Int. Conf. Comp.-Aided Des. (ICCAD)*, Nov. 2010.

[6] W. A. Dees, Jr. and P. G. Karger, "Automated rip-up and reroute techniques", *Des. Automation Conf. (DAC)*, pp. 432-439, 1982.

[7] C. J. Alpert et al., "What makes a design difficult to route", *Int. Symp. Physical Design (ISPD)*, pp. 7-12, Mar. 2010.