An Open Road Knows No Borders: The Contributions of UFRGS-UCSD Partnership to the OpenROAD Project

Vitor Bandeira[†*], Mateus Fogaça[†*], Jiajia Li[+], Eder Matheus Monteiro[*], Isadora Oliveira[†*], Ricardo Reis[†‡*] and Mingyu Woo[+]
†PGMicro/‡PPGC, *Instituto de Informática, Universidade Federal do Rio Grande do Sul
+CSE Department, UC San Diego, La Jolla, CA, USA
{vvbandeira, mpfogaca, emrmonteiro, isoliveira, reis}@inf.ufrgs.br, {ji1150, mgwoo}@ucsd.edu

***Abstract*—This paper outlines the tools developed by the UFRGS team in cooperation with UCSD to the OpenROAD project. We start showing ioPlacer, a tool for placement of IO pins in the core boundaries and its tight integration with OpenROAD's placement tool, RePlAce. Then, we describe the process of "open-sourcing" the academic global router FastRoute and the extensions required for the integration with a detailed router.**

I. Introduction

Around the world there are only a handful of research groups working on VLSI-CAD and related EDA tools—and even fewer companies. With advanced technology nodes and fabrication processes, constraints become harder to meet. Time closure, minimum frequency/delay, power dissipation, and energy consumption are some of the challenges that the EDA industry faces each and every day. As the complexity to design a chip increases, so does the cost of the underlying tool set needed to take this chip into fabrication. In this regard there are two main views, (i) EDA companies (i.e., those who sell tools) need to charge premium for their tools to offset the development and maintenance costs; and (ii) consumer-electronic industry (i.e., those who develop products) either need to scale their business to meet the throughput needed to turn a profit, or, more common for small businesses, fallback to COTS (commercially off-the-shelf) designs.

COST solutions can be a cheap alternative for small companies. However, these solutions leave a lot on the table w.r.t. performance and design space exploration. To reduce the cost of full custom designs, chip developers need access to tools that have a rich set of features, are reliable, scalable, and are affordable. Seeing this trend of fewer companies investing in buying/licensing EDA tools, and small companies diverting efforts towards COST designs, stakeholder of different spectrum's are combining forces. One example of this initiative is The OpenROAD project, lead and funded by the US Government through DARPA (Defense Advanced Research Projects Agency) in collaboration with the industry (i.e., Qualcomm, and Arm) and the Universities of California San Diego, Brown, Michigan, and Minnesota. The main goal of this project is to reduce the effort, expertise and the overall cost to design a full custom chip. The approach is to use technologies such as machine learning, design and problem partitioning, and a parallel e distributed architecture to leverage a 24h RTL-to-GDSII no human in the loop flow.

A key factor to reach such ambitious goals is open-source development, and thus collaboration. The adoption of an open-source development flow enables collaboration between entities geographically distant. UFRGS team already have had success with open-source EDA tools [2]. Rsyn framework and solid data models for physical design opened the doors for the partnership UFRGS-UCSD. Further, due to the OpenROAD flow need for a tool capable to perform I/O pin placement arose the first contribution of this partnership, i.e., the ioPlacer (Section II). The second contribution appears in the form of extending the existent global router FastRoute from the Iowa University to support commercial input format (i.e., LEF/DEF files), improve the already outstanding original code adding more features and user controllability of the tool—these were functionalities overlooked probably due to the fact that the original FastRoute targeted a contest and not widespread use in a complete flow such as the OpenROAD flow (Section III).

II. ioPlacer

In the OpenROAD flow, *ioPlacer* is the tool responsible for defining the locations of I/O pins in the core boundaries. ioPlacer receives as input (i) a metal layer for horizontal pins (left and right edges); (ii) a metal layer for vertical layers (top and bottom edges); and (iii) an initial placement. The output is an on-track position for each I/O pin in one of the given metal layers.[1,2]

The first step of ioPlacer computes all the possible I/O pin locations, called *slots* using tracks information. Then a Hungarian matching technique [3] is used to assign I/O pins to slots. The goal of the Hungarian matching is to minimize the total I/O nets half-perimeter wirelength (HPWL). Our Hungarian matching solver has a $O(n^3)$ runtime complexity, where n in this case is the number of candidate positions. We implement a divide-and-conquer strategy to avoid runtime issues in large floorplans. In our strategy, the core boundary is divided into equal-size regions, called *sections*. Then, I/O pins are assigned to sections and I/O pin placement is solved individually in each section.

It is well known in the literature that instances placement and I/O pin placement are interdependent [1][5]. To solve this chicken and egg problem, we propose the flow depicted in Figure 1 (b). Step 0 uses OpenROAD's global placement tool, RePlAce [9], to generate an initial instance placement.[3] In step 0, we disregard I/O nets since we do not know I/O pins locations yet. In step 1, we compute the slots and sections, as described in the previous paragraph. Step 2 performs I/O pin assignment considering the initial placement. Global placement is performed again, now considering I/O nets (Step 4). We iterate between steps 3 and 4 until HPWL converges.

Figure 2 shows the results of ioPlacer for a design with 1211 pins and 35K instances after the 1st and 10th iteration.

III. FastRoute

FastRoute4-lefdef is an open-source tool responsible for the global route in the OpenROAD flow. The algorithm base for FastRoute4-lefdef is from FastRoute 4 [6], the well-known open-source global router from Iowa State University. The underlying infrastructure comes from Rsyn [2].

[1] We only consider on-track locations to provide a router-friendly solution.
[2] Support for multiple horizontal/vertical layers is under development.
[3] Step 0 is not part of ioPlacer, i.e., is done by an external script.

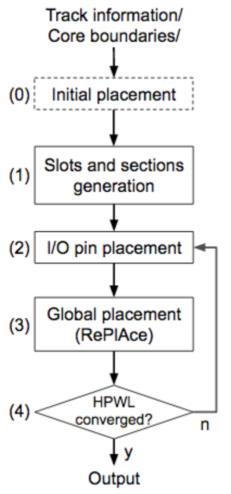
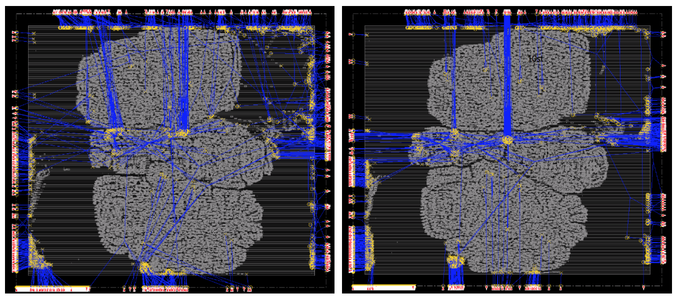


Fig. 1: ioPlacer flow.



(a) (b)

Fig. 2: Results of ioPlacer for a design with 1211 pins and 35K nets [4] after the 1st iteration (a) and 10th iteration (b).

FastRoute4-lefdef have support for LEF/DEF input files format. Given a placed design and its netlist, the percentage of reduction in the capacity of each edge in the global routing grid graph, and the minimum and the maximum routing layer available, our tool returns the global route for each net of the design, in the "guides" format established by the ISPD18 Initial Detailed Routing Contest [4].

Figure 3 shows the flow of FastRoute4-lefdef. Step 1 creates the global routing grid. The initial position of the grid is the lower bound of the die area. Each tile of the grid have the width and height defined as 15 $\times$ Metal 3 routing tracks spacing. Also, we create a grid for each metal layer of the design. Step 2 sets the capacities of the grid edges for each metal layer. The edge capacity of a layer is equal to the number of routing tracks of that layer that intersects the edge. Besides, we only compute capacities for the edges that match the preferred routing direction of the layer. Edges that do not match it have capacity set as zero. If a metal layer is blocked, i.e., it is not included in the between the minimum and maximum routing layers defined by the user, all edges of this layer will have the capacity set as zero.

Steps 3 and 4 compute adjustments in the capacities of each edge. In step 3, we translate the macroblocks and other obstacles of the design into reductions in the capacities of the edges. Only edges that intersect an obstacle of the design have a reduction in their capacities. Step 4 computes a global adjustment, according to a percentage of reduction received as an user-defined parameter. For each metal layer, all edges of the grid have a reduction of capacity, based on the current

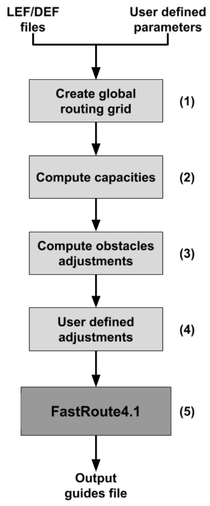


Fig. 3: FastRoute4-lefdef flow.

edge capacity and the reduction defined by the user.

Finally, Step 5 executes FastRoute 4 algorithm. We implemented an API to access FastRoute 4 functions from Rsyn infrastructure. The final step translates the original output format of FastRoute 4 into guides.

IV. Conclusion

This paper described the importance of an open-source RTL-to-GDSII flow, as the one proposed by the OpenROAD project, and the UFRGS contributions to it. The overall importance of this flow is to provide a complete framework, free of charge, to small companies and to researchers on the academic field. By doing this, the effort to design a full custom chip is reduced and, therefore, researchers can focus on the design space exploration. The UFRGS team contributed to this project by developing the ioPlacer, a tool that defines the position of I/O pins, and by extending FastRoute, a fast and efficient global router.

Another important aspect of this project is the world-wide contribution given by many different people and universities. This allows a better framework to be built by the exchange of knowledge and skills by so many researchers. The framework is still on expansion, and future works from the partnership UFRGS-UCSD include the integration flow test suite — which is currently under development.

Acknowledgments

The authors would like to thank Prof. Andrew B. Kahng and the ABK group for their guidance, technical support, and productive discussions on how to best implement our tools to be seamlessly integrated into the OpenROAD project flow.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq and FAPERGS.

Research at UCSD is supported by Qualcomm, Samsung, NXP Semiconductors, Mentor Graphics, DARPA (HR0011–18–2–0032), NSF (CCF-1564302) and the C-DEN center.

References

[1] A. Caldwell, A. B. Kahng, S. Mantik and I. L. Markov, “Implications of Area-array I/O for Row-based Placement methodology”, *Proc. IPDI*, 1998, pp. 93–98.

[2] G. Flach, M. Fogaca, J. Monteiro, M. Johann, and R. Reis, “Rsyn: An Extensible Physical Synthesis Framework”, *Proc. ISPD*, 2017, pp. 33–40.

[3] H. W. Kuhn, “The Hungarian Method for the Assignment Problem”, *Nav. Res. Logist.*, 2 (1955), pp. 83–97.

[4] S. Mantik, G. Posser, W.-K. Chow, Y. Ding, and W.-H. Liu, “ISPD 2018 Initial Detailed Routing Contest and Benchmarks”, *Proc. ISPD*, 2018, pp. 140–143.

[5] J. Westra and P. Groeneveld, “Towards Integration of Quadratic Placement and Pin Assignment”, *Proc. ISVLSI*, 2005, pp. 284–286.

[6] Y. Xu, Y. Zhang and C. Chu, “FastRoute 4.0: Global router with efficient via minimization”, *Proc. ASPDAC*, 2009, pp. 576–581.

[7] ioPlacer. https://github.com/The-OpenROAD-Project/ioPlacer

[8] FastRoute4-lefdef. https://github.com/The-OpenROAD-Project/FastRoute4-lefdef

[9] RePlAce. http://github.com/The-OpenROAD-Project/RePlAce

[10] The OpenROAD Project. https://theopenroadproject.org/