Understanding Negotiated-Congestion Routing for FPGAs

Mikhail Asiatici, Samuel Bosch, Zander Harteveld, and Stefan Nikolić

1 INTRODUCTION

Arguably the most important stage for timing closure in the *Field-Programmable Gate Array* (FPGA) CAD flow is placement. Most of the timing-driven placement algorithms strive to minimize some function of total connection criticality and delay [2]. Whereas in ASIC placement the final routed delay of a connection is highly correlated with its Manhattan length, the discrete nature of FPGA routing structure makes the Manhattan model very unrealistic. For that reason, individual connection delays are usually extracted by actually routing single connections with different tail and head locations. These delays are then stored in a look-up table which is indexed by the connection endpoints during the placement process [2]. This approach provides a much more realistic model than any founded only in geometry. However, in routing individual connections, the phenomenon of *congestion*, occurring when optimum routes of two or more connections use the same routing tracks, forcing one (some) of them to be diverted, is entirely ignored. Although the *negotiated-congestion* routing algorithms, which have been the state-of-the-art for almost 30 years, do try to implement the critical connections using the shortest possible routes, in practice, the discrepancy between the delays predicted using the place-time model that ignores congestion and the final routed delay can reach several percent of the critical path delay, which is significant by modern standards. Of course, simultaneously solving the placement and the routing problem is infeasible for today's problem sizes, but a better understanding of what causes this discrepancy and why it is more pronounced for some circuits and less for others can enable us to create better models that will lead to either better solution quality, or reduced runtime, as well as give important insights into architectural modifications that could allow for a tighter integration of placement and routing. The purpose of this project is to harness the methods of network science and data analysis to deepen this understanding that currently appears not to reach much further than mere intuition with no quantitative support.

2 REPRESENTATION

We can construct a bipartite graph where each net of the circuit to be placed on the FPGA represents a node in one partition, while each routing track of the FPGA is represented by a node in the other (Figure 1). Because the router does not exit the bounding box of any net by more than a fixed amount, the graph will be rather sparse. The weight of an edge between the net u and the routing track v should describe how likely it is that the final solution will be of good quality if the net u is routed through the track v. We can calculate the distance $d(u, v)$ as the sum of the delays of the shortest paths between the net's source and each of its targets, traversing the track v. Each of the addends in the sum is weighted by the criticality of the corresponding connection, normalized by the highest criticality among all the connections of the net. This is illustrated in Figure 2. The inverse of the distance can be used to weight the corresponding edge. We also need to weight the net-representing nodes, in proportion to their criticality.

3 THE MAIN QUESTION

Let us start from a loose conjecture.

Conjecture 1. *The routed delay will be noticeably larger than the post-placement prediction iff there are heavy clusters in the projection of the net-representing partition.*

Careful choice of the weights of edges in the projection is crucial: if two nets share a potentially used track, it does not mean that an optimum routing solution can not be achieved without both of them using that track. Let us designate the weights in the original bipartite graph as $w(u, v)$ and those in the projection as $w_p(u, v)$. Let us also choose two nets u_1 and u_2 and designate their neighborhoods in the bipartite graph as N_1 and N_2, respectively. Then we can define $w_p(u_1, u_2)$ as follows:

$$\langle w_c(u_1)\rangle = \frac{\sum_{v\in\{N_1\cap N_2\}} w(u_1, v)}{|N_1 \cap N_2|} \tag{1}$$

$$\langle w_{uc}(u_1)\rangle = \frac{\sum_{v\in\{N_1\setminus N_2\}} w(u_1, v)}{|N_1 \setminus N_2|} \tag{2}$$

$$w_p(u_1, u_2) = max(\{\langle w_c(u_1)\rangle - \langle w_{uc}(u_1)\rangle, \langle w_c(u_2)\rangle - \langle w_{uc}(u_2)\rangle\}) \tag{3}$$

Finally, we can drop the negative edges. Defined in this way, the weights tell us how much a net would lose by giving up its preferred resources to its opponent. Testing the conjecture now reduces to clustering the projection, weighing the clusters and comparing the weights to an appropriate threshold. A potentially good definition for a cluster weight could be the following:

$$W = \frac{\sum_{(u,v,w_p)\in E} uvw_p}{|E|} \tag{4}$$

u and v in the expression designate the weights of the corresponding nodes and E is the set of edges contained in the cluster.

4 ARE THE HEAVY CLUSTERS INHERENT OR CREATED BY THE PLACER?

We can first cluster the circuit graph itself and observe if the nodes of the heavy cluster are members of the same cluster of the circuit as well. If that is the case, we can have some confidence that it is not the placer that should be blamed. We can also embed the circuit graph in a plane, using a standard technique such as t-SNE, and observe the proximity of the nodes forming the heavy cluster. This experiment could help us better understand the natural structure of circuits and perhaps realize how the architecture could be optimized to better suit it, or how the algorithms could be made to take advantage of it.

5 TECHNICAL DETAILS

We will use the MCNC [3] benchmark set and a simple FPGA architecture with all tracks spanning four logic blocks. VTR [1] will provide the placement and routing algorithms. Some modification of its source code will be necessary to extract the appropriate data.

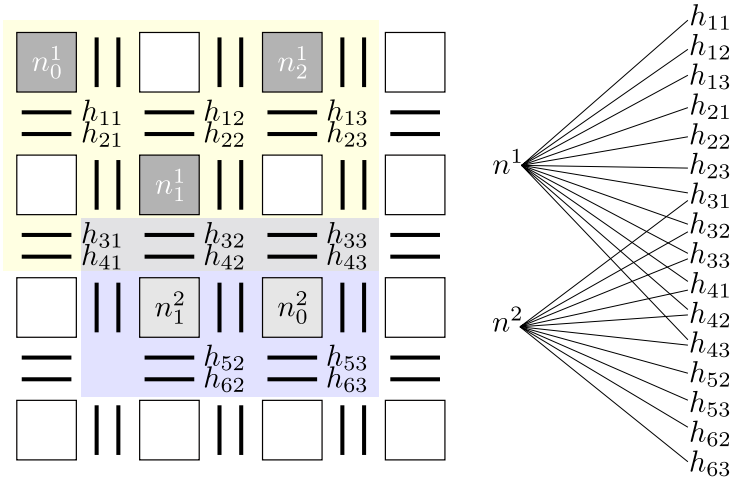

Figure 1: An example of construction of the bipartite graph describing the relations between the nets of a placed circuit and the FPGA's routing tracks. For clarity, only two nets are shown, and only horizontal segments are labeled and represented in the graph.

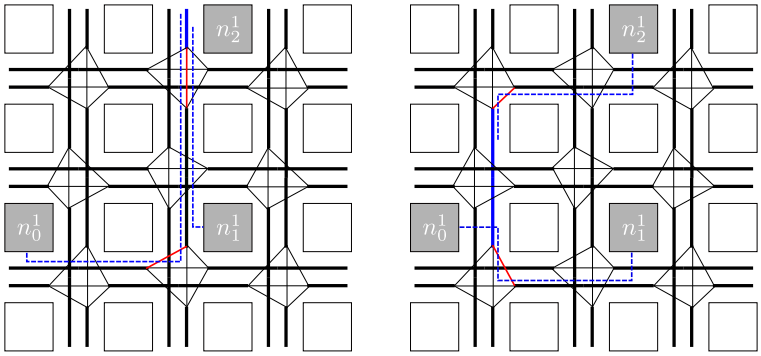

Figure 2: An example of edge-weight calculation. Thin black lines represent the connectivity achievable by the switch boxes. Here we assume that the delay is proportional to the number of switch boxes crossed by a connection. The track v for which we are computing the weight is highlighted in blue. In the left figure, two switch boxes (red) are traversed to reach v; then, to reach the target n_1^1, we need to cross one more switch box, while n_2^1 is immediately accessible. Hence the distances to n_1^1 and n_2^1 through v are 2 and 3, respectively. Assuming that criticalities of (n_0^1, n_1^1) and (n_0^1, n_2^1) are 0.7 and 0.9, respectively, we obtain the weight of $\frac{1}{3.9}$. In the figure on the right, both targets are one switch box away from v, and v is reachable from the source without traversing any switch boxes, so the weight is $\frac{1}{1.6}$. In practice, of course, the real router is used to obtain the paths, eliminating any need for approximation of the delay by hops.

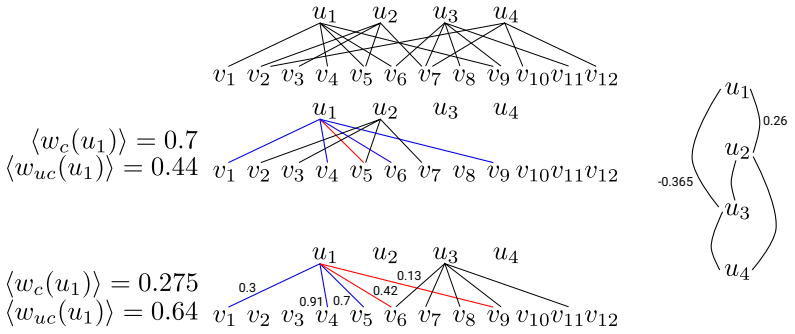

Figure 3: An example of determining the edge weights in the net-representing partition projection. Averages are only computed for the net u_1. Assuming that u_1 has the dominant difference between the average congested and uncongested weights, the projection weights will be as annotated in the graph on the right. Due to its negative weight, the edge (u_1, u_3) does not appear in the final projection.

REFERENCES

[1] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose, and V. Betz. VTR 7.0: Next generation architecture and CAD system for FPGAs. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 7(2):6:1–6:30, June 2014.

[2] A. Marquardt, V. Betz, and J. Rose. Timing-driven placement for fpgas. In *Proceedings of the 2000 ACM/SIGDA Eighth International Symposium on Field Programmable Gate Arrays*, FPGA '00, pages 203–213, New York, NY, USA, 2000. ACM.

[3] S. Yang. Logic synthesis and optimization benchmarks user guide, version 3.0. Technical report, Microelectronics Center of North Carolina, Research Triangle Park, N.C., Jan. 1991.