The Effects of Net Ordering in Global Routing

Arthur Bianco
Dept. of Computer Science and Statistics (INE)
Federal University of Santa Catarina (UFSC)
Florianopolis, Brazil
a.p.bianco@grad.ufsc.br

Renan Netto, Tiago Fontana, Sheiny Fabre,
José Luis Güntzel
Dept. of Computer Science and Statistics (INE/PPGCC)
Federal University of Santa Catarina (UFSC)
Florianopolis, Brazil
{renan.netto, tiago.fontana, sheiny.fabre}@posgrad.ufsc.br,
j.guntzel@ufsc.br

***Abstract*—The global routing step is critical to ensure a good detailed routing solution, and therefore the feasibility of the entire circuit. Techniques for global routing either operate directly on a 3-D grid or flatten the routing space into a 2D grid, looking to speed up the routing step. In the latter case, the 2D solution is then projected back onto the 3D space by the so-called layer assignment step. Whichever method is employed, many routing techniques rely on maze and/or pattern routing, which operates on a net-by-net basis using as input a netlist sorted according to some parameters. In this work, we analyze various sorting methods and their effects on the global routing outcome. Particularly, we found that sorting the netlist in the ascending order could produce results, on average, with 5.91% fewer vias than in the descending order.**

***Index Terms*—VLSI, EDA, global routing, net sorting**

I. Introduction

To cope with the ever-increasing complexity in IC design, modern EDA solutions divide the routing step into global and detailed routing. In global routing, the chip area is commonly represented as a coarse grid graph. The nodes are called gcells, which represent rectangular sections of the circuit. Routing resources are represented by capacity-bearing edges that are created between adjacent gcells [1]. The nets are then routed in terms of gcells, creating a general path for the detailed routing stage to follow.

A well-known approach to the global routing problem consists of routing each net entirely in a 3D grid using a maze router (e.g., [2]). This method produces a good solution quality at the cost of very high execution time. Another common approach for routing the gcells is to flatten the 3D routing space into a 2D grid, as done by FastRoute 4.0 [3], NTHU-Route 2.0 [4], MaizeRouter [5], and then route the nets all on the same layer through maze or pattern routing. After that, the routed segments are projected back onto the 3D grid through layer assignment. This approach is known to produce good results at reasonable run times. Archer [6] implements its solution on both 3D and projection-based approaches, and its authors state that for the benchmarks tested, the advantages of the projection-based technique outweigh its disadvantages.

CU-GR [7], however, proposed a pattern routing and layer-assignment initial routing, followed by a maze routing phase. This solution is novel in that it performs all of the steps directly in the 3D space, yet displaying excellent run time performance with contest-winning [8] solution quality.

Routing solutions based on maze routing and pattern routing algorithms operate on a net-by-net basis and therefore the quality of their outcome strongly depends on the order through which the nets are processed. The authors of [6] proposed an improvement to an earlier version of their router and reported that the most notable improvement to the layer assignment algorithm was sorting the nets in ascending order of their HPWL (half perimeter wirelength). NTHU-Route 2.0 [4] improved NTHU-Routes' [9] solution by reversing the rip-up and reroute (RRR) order to happen according to descending order of net bounding box size and reported better wirelength. CU-GR [7], interestingly enough, orders the nets in the ascending order of HPWL, before pattern routing and before each RRR iteration.

This paper aims to analyze the effects of net ordering during the routing steps. Geared towards that, we relied on CU-GR and altered the sorting of the nets, before the pattern routing phase, and in each RRR iteration. We tested ascending and descending orders on different ordering parameters which include HPWL and pin count. We evaluated the sorting methods by measuring their impact on the total wirelength, number of vias, and number of nets in the first RRR iteration. We chose CU-GR because it is an open-source state-of-the-art global router that allows for changing the routing order.

The remainder of this paper is organized as follows. Section II presents the relevant features of CU-GR used in this study. We give the methodology for the tests conducted in section III. The results are presented in section IV and the conclusion and future work are shown in section V.

II. The Router

CU-GR was the winner of the ICCAD'19 global routing contest [8]. The contest's problem proposition was to create a global router that, given a DEF and a LEF file, should output the circuit routing guides of a valid solution. Then, the resulting files would be fed to Dr. CU [10], an open-source detailed router that won the ISPD19 Contest [11], to generate the detailed routing solution that would be used for scoring purposes.

Figure 1 depicts the flow of CU-GR's algorithm. It has 3 stages: Initial Routing, Multi-level 3D Maze Routing, and Route Guide Generation and Patching.

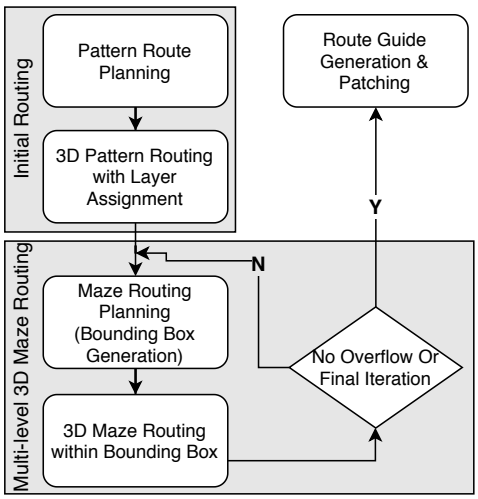

Fig. 1. Overall flow for CU-GR's algorithm. Image adapted from [7].

A. Initial Routing

First, FLUTE [12] is used to create the rectilinear Steiner minimum tree (RSMT) for each multi-pin net. Then, a technique named edge shifting [13] is applied, where edges of each RSMT are shifted away from each other to reduce congestion. Next, the order between segments of the same net is decided by breaking the nets down into 2 pin nets by a depth-first search traversal through the segments previously generated. Subsequently, the nets are sorted in the increasing order of their HPWL and a dynamic programming solution is employed to perform the 2D pattern routing together with the layer assignment.

B. Multi-level 3D Maze Routing

Now that the initial routing is complete, the nets with violations will undergo RRR iterations. Maze routing is carried out in a coarsened gcell grid, where gcells are compressed into blocks, so as to find an initial routing solution at a quicker pace. Then, another maze router will perform the routing within the bounding boxes created by the previous stage to find an ideal solution in terms of the original gcells. This technique was developed as a means of reducing the 3D maze routing space while still not giving up too much in solution quality.

C. Route Guide Generation and Patching

As a post-processing step, CU-GR adds patches of guides around identified congestion-intensive regions. If regions directly above any pins are identified as lacking in routing resources, the router will patch guides above that pin. Guide segments are also added around segments that are considered long enough or regions with inevitable routing violations.

III. METHODOLOGY

We altered CU-GR's sorting procedure to implement 6 sorting methods different from the original one. Thus, the 7 investigated sorting methods were: nets sorted in ascending order of their HPWL (ascendingHP), which corresponds to the original CU-GR sorting method, nets sorted in descending order of their HPWL (descendingHP), nets sorted in ascending order of the sum of HPWL and pin count (ascendingNPinsHP), nets sorted in descending order of the sum of HPWL and pin count (descendingNPinsHP), nets sorted in ascending order of pin count (ascendingNPins), nets sorted in descending order of pin count (descendingNPins), nets in the order in which they were found in the def file (unordered).

We ran CU-GR, with a single thread, on a subset of the benchmarks provided for the ICCAD'19 global routing contest [8]. We chose these circuits, in particular, because they were the same ones chosen by the contest organizers to evaluate and score the routers. The benchmark set is composed of six circuits, each one in two versions: one with 5 routing layers and another with 9 routing layers. The versions with 5 layers are marked with "metal5" and have the same netlist and placement as their homonymous counterparts. Each benchmark has a net count between 72 thousand and 900 thousand. We chose these sorting methods because they use important net attributes that are commonly used to create an efficient sorting method. We then gathered the data reported by the router, to be used in our analyses. For reproducibility purposes, the source code and scripts used in this work are available at [14].

IV. RESULTS

Table I allows for comparing the global routing results for the ICCAD19 benchmarks using the 7 sorting methods: Total estimated wirelength (TWL), via count (#vias), and the number of nets ripped up and rerouted (#RRR) on the first iteration of multi-level 3D maze routing is reported. The CU-GR original sorting method (ascendingHP) is highlighted in yellow and its results were used as a baseline to present the results of the other sorting methods. The benchmarks with "19" in their names are larger than those with "18". The circuits increase in size as the second number in their names increases, which means the largest benchmark is "19test9" (and "19test9metal5") and the smallest is "18test8" (and "18test5metal5").

It is worth noting the relevance of the circuit attributes reported in table I. Due to their parasitic capacitances and resistances, circuit wires have a significant impact on both delay and power and thus, their lengths must be minimized. Particularly, vias have a significantly higher RC value than metal wires and therefore should also be minimized [15]. According to [6], 50% of the total runtime is spent on maze routing, even though only 2% of the nets are routed using a maze router. Therefore it is also important to reduce as much as possible the number of nets requiring RRR iterations, to reduce runtime.

From the results in table I, one can observe that for the majority of descending cases the CU-GR original method (ascendingHP) led to bigger values of TWL for the three smallest circuits and to smaller values of TWL for the three largest ones, albeit the difference is less than 1% in all cases. On the other hand, ascendingHP led to the smallest number of vias (#vias) for all cases. Concerning #RRR, the original

TABLE I
COMPARISON OF GLOBAL ROUTING RESULTS ACROSS DIFFERENT BENCHMARKS AND SORTING METHODS

Name	18test5			18test8			18test10			19test7		
	TWL	#vias	#RRR	TWL	#vias	#RRR	TWL	#vias	#RRR	TWL	#vias	#RRR
ascending HP	2.70E+07	8.56E+05	8.68E+02	6.44E+07	2.18E+06	2.84E+03	6.68E+07	2.31E+06	5.83E+03	1.19E+08	3.13E+06	5.33E+03
descending HP	-0.02%	+4.90%	-16.71%	-0.25%	+5.99%	-24.59%	-0.20%	+6.86%	-9.46%	+0.46%	+6.63%	+6.73%
ascending NPinsHP	-0.03%	+0.19%	+1.15%	0.00%	0.00%	0.00%	+0.03%	+0.15%	-0.98%	+0.01%	+0.34%	+3.49%
descending NPinsHP	-0.04%	+4.36%	-18.78%	-0.30%	+5.20%	-25.15%	-0.23%	+5.99%	-10.15%	+0.34%	+5.44%	-2.10%
ascending NPins	-0.01%	+1.90%	-5.53%	-0.13%	+2.24%	-15.44%	+0.06%	+2.14%	-1.58%	+0.03%	+2.46%	-1.37%
descending NPins	-0.04%	+1.91%	-22.81%	-0.22%	+1.86%	-19.38%	-0.23%	+2.74%	-8.00%	+0.08%	+0.86%	-22.22%
unordered*	-0.05%	+1.89%	-7.37%	-0.23%	+2.32%	-19.66%	-0.11%	+2.96%	-4.00%	+0.06%	+2.12%	-12.64%

Name	19test8			19test9			18test5metal5			18test8metal5		
	TWL	#vias	#RRR	TWL	#vias	#RRR	TWL	#vias	#RRR	TWL	#vias	#RRR
ascending HP	1.82E+08	5.75E+06	2.73E+03	2.74E+08	9.60E+06	2.42E+03	2.79E+07	8.03E+05	1.14E+04	6.47E+07	1.96E+06	3.43E+04
descending HP	+0.45%	+5.35%	+2.67%	+0.56%	+5.33%	+72.73%	-0.54%	+5.58%	-32.66%	-0.21%	+5.97%	-24.10%
ascending NPinsHP	-0.02%	+0.17%	-5.49%	+0.02%	+0.19%	+4.92%	+0.05%	+0.21%	-2.27%	-0.08%	+0.08%	-1.29%
descending NPinsHP	+0.39%	+4.70%	-8.34%	+0.54%	+4.66%	+61.86%	-0.49%	+5.41%	-32.13%	-0.18%	+5.76%	-24.03%
ascending NPins	+0.05%	+1.86%	-1.79%	+0.04%	+1.88%	+33.22%	-0.14%	+1.14%	-10.69%	-0.10%	+0.86%	-7.31%
descending NPins	+0.19%	+1.90%	-22.93%	+0.29%	+1.78%	+7.89%	-0.13%	+4.01%	-15.24%	+0.07%	+4.25%	-12.30%
unordered*	+0.14%	+1.96%	-6.69%	+0.19%	+1.78%	+51.53%	-0.12%	+2.56%	-15.08%	-0.07%	+2.72%	-11.77%

Name	18test10metal5			19test7metal5			19test8metal5			19test9metal5		
	TWL	#vias	#RRR	TWL	#vias	#RRR	TWL	#vias	#RRR	TWL	#vias	#RRR
ascending HP	7.09E+07	2.08E+06	3.88E+04	1.07E+08	3.07E+06	2.24E+04	1.78E+08	5.60E+06	3.13E+04	2.68E+08	9.34E+06	4.30E+04
descending HP	-0.15%	+5.71%	-19.85%	+0.13%	+5.80%	-9.96%	+0.15%	+6.36%	-27.88%	+0.15%	+6.42%	-26.21%
ascending NPinsHP	+0.06%	+0.04%	-0.06%	+0.03%	+0.27%	+1.17%	+0.04%	+0.12%	-1.05%	+0.03%	+0.09%	+1.34%
descending NPinsHP	-0.13%	+5.52%	-20.27%	-0.02%	+5.01%	-13.89%	+0.09%	+5.88%	-30.20%	+0.10%	+5.89%	-29.11%
ascending NPins	-0.03%	+0.67%	-3.95%	-0.03%	+2.13%	-2.59%	-0.08%	+1.57%	-11.16%	+0.02%	+1.47%	-3.51%
descending NPins	0.12%	+4.29%	-9.33%	-0.21%	+1.45%	-17.02%	+0.02%	+3.50%	-25.07%	+0.02%	+3.48%	-24.58%
unordered*	0.06%	+2.58%	-7.19%	-0.18%	+1.81%	-10.70%	0.00%	+2.65%	-17.58%	-0.04%	+2.48%	-15.32%

method was outperformed by at least 5 of the 6 other methods for all circuits, excluding 19test7 and 19test9.

For each pair of ascending and descending methods, the TWL of the descending case is usually less than 1% lower than its ascending counterpart. It seems that the smaller the benchmark the more apparent this pattern is, and with bigger benchmarks this behavior switches. On the other hand, the difference in #vias is more pronounced and generally higher for the descending case. When looking at #RRR, the number of nets that require it is lower for the descending cases, except for 19test7, 19test8 and 19test9.

In comparing both the HPWL against the NPins sorting methods and the HPWL against the NPins + HPWL, all in the ascending order, we find that in a few cases there were reductions in TWL averaging at -0.06% (min=-0.01%, max=-0.14%), while in other cases there were inscreases with an average of 0.04% (max=+0.06%, min=+0.01%), but all were not significant. However, the #vias is higher for the NPins (avg=1.74%, max=2.74%, min=0.67%) and the NPins + HPWL methods, (avg=0.15%, max=+0.34%, min=+0.00%). The unordered case is usually comparable, for every benchmark, to the NPins cases, when looking at any of the data reported.

In order to evaluate how the different sorting methods impacted on each step of the routing algorithm, we analyzed the #vias and TWL after each stage for the circuits. Fig 2, breaks down the #vias throughout the routing steps for the 18test8 benchmark. It shows that at the end of the pattern route step, the #vias for the descending methods is higher than that of the ascendingHP method. Even though the first RRR iteration (RRR1) drops the #vias for all cases, the descending methods still have a much higher #vias than the ascendingHP method, and subsequent RRR iterations do not further reduce

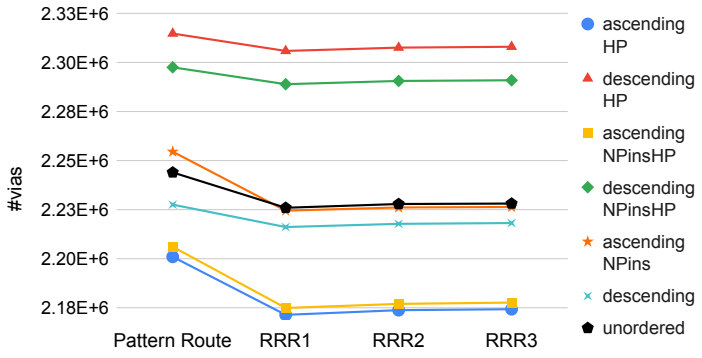

Fig. 2. #vias for benchmark "18test8" after pattern route and after every RRR iteration.

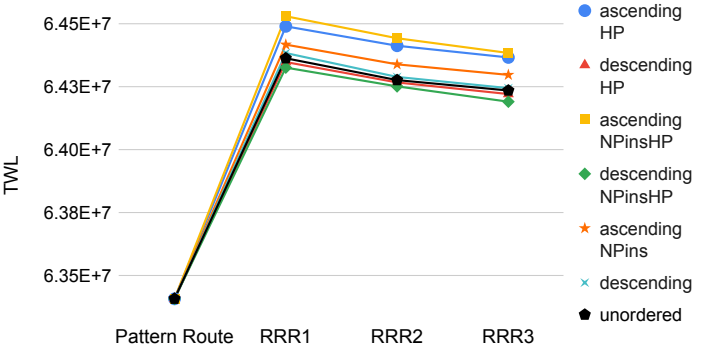

Fig. 3. TWL for benchmark "18test8" after pattern route and after every RRR iteration.

the #vias. Fig 3 shows the TWL throughout the routing steps. We see that the TWL at the end of the pattern route phase is virtually the same for all sorting methods. It is only after the first RRR iteration that we can observe a difference between each method. Next, all the methods undergo a reduction and after the third RRR iteration (RRR3), the TWL is slightly lower for the descending methods, however, it is not low enough to compensate for the higher via usage.

We note that for the 3 largest benchmarks, when using the descending methods, not only the #vias increased but also the TWL. This is in contrast with the 3 smaller benchmarks, where TWL is lower for the descending methods. We believe this happens due to the RRR iterations also being sorted in the same manner: when ripping up and rerouting the larger nets first, the smaller nets undergo long detours to find a valid route, further increasing wirelength.

Our hypothesis for the increased number of vias when sorting the nets in descending order of HPWL is that when routing the larger nets first they often take up routing resources in lower layers, rather than higher, to benefit from a reduced via usage. Therefore, the router will struggle to find space for shorter nets to access the pins, causing complicated detours and therefore increasing via usage. In contrast, the larger nets will have plenty of long unoccupied tracks to be routed on.

In addition, routing long planar segments in upper layers requires fewer vias than assigning several short segments to those upper layers [6]. As a result, routing the shorter nets first reduces the overall number of vias, especially because the majority of nets in the circuit are short. Figure 4 plots a histogram of nets by their HPWL, and shows that 80% of the nets have up to 16 of HPWL. A similar behavior can also be observed in every other benchmark we used.

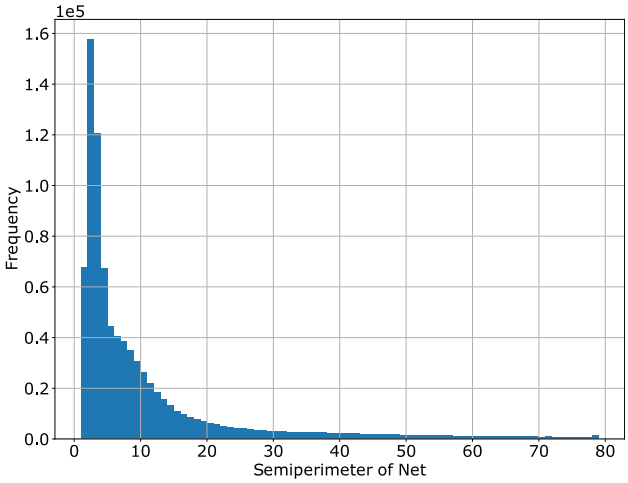

Fig. 4. Net semiperimeter histogram for benchmark "19test9". The X-axis represent the half-perimeter sizes the nets can fall into and the Y-axis counts the number of nets in each category.

V. Conclusion and Future Works

In this paper, we compared the outcome of IC global routing when 7 distinct sorting methods are applied in net-by-net stages. Particularly, we evaluate the resulting total estimated wirelength, number of vias, and number of nets requiring rip up and reroute after an initial stage of pattern routing, followed by subsequent iterations of maze routing. Sorting the nets by their HPWL in an ascending order yields the best results in terms of #vias and TWL. However, sorting in the opposite sense does not necessarily give a lower total estimated wirelength usage, but a higher via usage. The average via count across the benchmarks of the descendingHP method is 5.91% higher than that of the ascendingHP method. Regarding the #RRR, the original sorting method displays worse performance than at least one other sorting method for all benchmarks tested, except for 19test9.

Future research on this subject could take place by sorting the nets using different constraints such as timing or congestion information. Further, It would be worth performing these and other tests with the detailed router and reporting the real definitive score, for completion and confirmation of the results.

References

[1] Andrew B. Kahng, Jens Lienig, Igor L. Markov, and Jin Hu. *VLSI Physiscal Design: From Graph Partitioning to Timing Closure*. Springer Science+Business Media, 2011.

[2] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(6):1066–1077, 2008.

[3] Yue Xu, Yanheng Zhang, and Chris Chu. Fastroute 4.0: Global router with efficient via minimization. In *2009 Asia and South Pacific Design Automation Conference*, pages 576–581, 2009.

[4] Y. Chang, Y. Lee, J. Gao, P. Wu, and T. Wang. Nthu-route 2.0: A robust global router for modern designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(12):1931–1944, 2010.

[5] M. D. Moffitt. Maizerouter: Engineering an effective global router. In *2008 Asia and South Pacific Design Automation Conference*, pages 226–231, 2008.

[6] Muhammet Mustafa Ozdal and Martin DF Wong. Archer: A history-based global routing algorithm. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(4):528–540, 2009.

[7] Jinwei Liu, Chak-Wa Pui, Fangzhou Wang, and Evangeline FY Young. Cugr: Detailed-routability-driven 3d global routing with probabilistic resource model.

[8] S. Dolgov, A. Volkov, L. Wang, and B. Xu. 2019 cad contest: Lef/def based global routing. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–4, 2019.

[9] Jhih-Rong Gao, Pei-Ci Wu, and Ting-Chi Wang. A new global router for modern designs. In *2008 Asia and South Pacific Design Automation Conference*, pages 232–237, 2008.

[10] Gengjie Chen, Chak-Wa Pui, Haocheng Li, Jingsong Chen, Bentian Jiang, and Evangeline FY Young. Detailed routing by sparse grid graph and minimum-area-captured path search. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 754–760. ACM, 2019.

[11] Wen-Hao Liu, Stefanus Mantik, Wing-Kai Chow, Yixiao Ding, Amin Farshidi, and Gracieli Posser. Ispd 2019 initial detailed routing contest and benchmark with advanced routing rules. In *Proceedings of the 2019 International Symposium on Physical Design*, pages 147–151, 2019.

[12] C. Chu and Y. Wong. Flute: Fast lookup table based rectilinear steiner minimal tree algorithm for vlsi design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(1):70–83, 2008.

[13] M. Pan and C. Chu. Fastroute: A step to integrate global routing into placement. In *2006 IEEE/ACM International Conference on Computer Aided Design*, pages 464–471, 2006.

[14] Cu-gr github fork. https://github.com/RamAddict/cu-gr/tree/SForum.

[15] Laung-Terng Wang, Yao-Wen Chang, and Kwang-Ting Cheng. *Electronic Design Automation: Synthesis, Verification, and Test*. Morgan Kaufmann Publishers, 2008.