A local congestion elimination technique driven by overflow

Wei Wu\(^1\)\(^,\) Zhixiong Di\(^1\)\(^,\)\(^a\), Jiangyi Shi\(^2\), Quanyuan Feng\(^1\), and Zhengguang Tang\(^2\)

Abstract In the physical design of VLSI circuits, the congestion generated in placement stage tends to enlarge the total wirelength (TWL) and further worsens the timing and routability. In this letter, a local congestion elimination technique is proposed which can be compatible with available commercial P&R EDA tools. Driven by overflow value, the optimal keep-out margins being added around the highest pin cells in specific congestion regions are searched using simulated annealing (SA) algorithm and ant colony optimization (ACO) algorithm respectively to ameliorate local congestion. Experimental results have shown that the proposed technique can reduce the design rule violations (DRV), short and TWL significantly.

Keywords: design automation, physical design, placement, congestion, overflow, heuristic algorithm

Classification: Integrated circuits (memory, logic, analog, RF, sensor)

1. Introduction

As the feature size of semiconductor process technology nodes further scales down, the industry is greatly challenged in terms of manufacturing and design [1]. Routing congestion in a VLSI design may cause unroutability or make large delays due to detoured long wires, especially for large and congested circuits [2]. In modern physical implementation of VLSI circuits, routability has been a primary concern even more important than timing [3, 4]. Consequently, it is vital to ameliorate congestion during the placement stage. The mainstream methods for congestion handling can be classified into two types: congestion estimation and congestion elimination.

There have been several pioneering works laying emphasis on the congestion estimation, since the early prediction in the physical design flow is critical to reduce design turnaround time and cost [5], and can avoid generating an unroutable design. Several recent works devote to studying prediction model by concentrating on the factors that affect congestion [6, 7, 8, 9]. Others use probabilistic congestion estimation techniques [10, 11]. Although these methods can improve the estimation accuracy of congestion, there is still a gap between these models and the actual detailed routing.

To narrow this gap, machine learning has been applied in this field. Machine learning has become pervasive in various research fields and commercial applications, and achieved satisfactory products [12]. Supervised learning is employed to detect detailed routing violations [13, 14, 15]. A machine learning framework is proposed to predict detailed routing short violations just from a placed netlist [16]. In summary, these approaches can further improve the estimation accuracy and robustness. However, there are a large number of congestion features have to be extracted, increasing the complexity of the prediction.

In terms of congestion elimination methods, several recent works have explored state-of-the-art global routers to improve routing quality and decrease congestion [17, 18, 19, 20, 21, 22, 23, 24, 25, 26], by iteratively rippling up and rerouting overflowed nets using maze routing. Although the optimization effect is significant, maze routing algorithm is slow and time-consuming. To address the shortcoming, the maze routing is replaced by other improved routing algorithm to realize a highly fast global router [27]. Achieving routing parallelism is the other solution to improve runtime of routing process [28, 29]. Another method is to apply keepout margins around the high-pin cells to provide more space to relieve congestion [30].

In this letter, we propose a congestion elimination technique based on heuristic algorithms driven by overflow value. Compared with the previous work [30], there are two main differences: (1) In [30], the cells added keepout margins around are complex cells with 4 or more pins, but in our work, the targets are characterized by the maximum pin number and total pin number. This selection mechanism is more targeted and can reduce cell number. (2) The keepout margins in [30] are only added to the left and right sides of each cell, and their width is always 1 unit. In some severe congestion situations, the width value is not large enough to reserve more routing space. So, in our work, not only the keepout margins are added to 4 sides of each cell, but also two heuristic algorithms, simulated annealing (SA) and ant colony optimization (ACO) algorithms, are respectively explored to adjust the appropriate keepout margin width. Our contributions are as follows:

1) A congestion-density-based region merging technique is presented to find the most congested region, and a selection mechanism of high-pin cells is proposed.

2) Simulated annealing and ant colony optimization algorithms are respectively explored to calculate the optimal width of keepout margins added around high-pin cells.

2. Proposed method

Congested region is prone to emerging around the high-pin cell which is referred to as HPC. Moreover, congestion will

---

1 The School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China
2 The School of Microelectronics, Xidian University, Xi’an 710071, China
a) zxdi@home.swjtu.edu.cn

DOI: 10.1587/elex.17.20200232
Received July 3, 2020
Accepted July 27, 2020
Publicized August 14, 2020
Copyedited September 10, 2020

Copyright © 2020 The Institute of Electronics, Information and Communication Engineers
emerge as an extremely knotty issue especially under the situation where many HPCs are placed centrally. Therefore, HPC is one of the decisive factors ought to be optimized.

For the reason above, setting keepout margins around the HPCs to push their adjacent cells away is an effective way to relieve local congestion. The process is shown in Figure 1, in which more space will be reserved around the cell-3 for routing after adding a keepout margin.

Our method is mainly composed of three stages: firstly, find the specific congested region which is called HCD-region; Then count the HPCs in that region; Finally, calculate the appropriate width of keepout margins added around HPCs by SA and ACO algorithms respectively.

### 2.1 HCD-region and HPC selection

An oversized congested region will comprise of numerous HPCs, which will increase chip’s cell density with the addition of many keepout margins. Conversely, too small congested region is hardly able to alleviate congestion effectively due to insufficient optimized HPCs. In order to obtain the appropriate size of congested region, an evaluation representation used for assessing congestion severity of congested region should be defined.

Overflow, defined in equation (1), is equal to the number of nets crossing the global routing cell (GRC) edge $N_{net}$ minus the number of available routing tracks $N_{track}$. It is frequently used to evaluate the severity of congestion and demonstrate the congestion circumstance in EDA tools. Thus, we focus our attention on GRCs with high overflow value. The congestion density ($CD$) is defined in equation (2). $CD$ is the ratio of the total number of GRC with high overflow value to the area $S$ of congested region where these GRCs are located.

$$Overflow = N_{net} - N_{track} \quad (1)$$

$$CD = \frac{N_{GRC}}{S} \quad (2)$$

Now that $CD$ is defined above, a HCD-region representing high congestion density region is obtained as exhibited in Figure 2. Key points of this mechanism are as follows:

1) A default overflow threshold is set, the GRC whose overflow value is higher than it will be picked out.

2) Expand all selected GRCs’ coordinates around by one row height to get sub-regions.

3) Since overlaps might exist among expanded sub-regions, $CD$ change is detected to decide whether overlapping sub-regions merge. If $CD$ of the merged region rises compared with that of the unmerged ones, the merged region will be retained, otherwise it will be abandoned.

4) The region with highest $CD$ is selected as HCD-region, in which all HPCs will be set keepout margins around to optimize congestion.

After determining a HCD-region, we count the number, pin number and total pin number of each standard cell in the specific HCD-region, and select the object to be optimized which has the maximum total pin number in the cells with maximum pin number.

### 2.2 Calculation of the keepout margin width

Undersized keepout margins fail to reserve sufficient routing resources, conversely although oversized keepout margins are capable of diminishing local congestion, they excessively squeeze the adjacent GRCs and are likely to cause more DRV instead. Thus two heuristic algorithms, the SA and ACO algorithms, are applied separately to calculate the suitable width of keepout margins.

1. Simulated annealing mode: The core strategy of SA algorithm is to calculate an acceptance probability to accept a solution worse than the current one. There will be a certain probability to jump out of the local optimal solution and achieve a global optimal solution. Therefore, SA algorithm is appropriate for the calculation of the optimal size of keepout margins.

The Algorithm 1 is adopted to search the width of keepout margins based on SA mechanism. The proposed method starts with an initial width $W_0$ which is usually set to less than one row height. $T$ represents the annealing times, $\alpha$ denotes the step rate smaller than 1, and $k$ is Boltzmann coefficient.

During the process, if the current overflow value is less than the last one, save the placement and adjust width based on equation (3). Conversely, if larger, an acceptance probability $P$ will be calculated according to equation (4), in which $R$ denotes the growth rate relative to last overflow. Boltzmann coefficient $k$ is equal to 0.7214 calculated at the situation where acceptance probability is 50% when overflow value is increased by 50%. If $P$ is greater than a random number $r \in [0, 1)$, continue adjusting the width, otherwise stop calculation and choose the placement with lowest overflow as best results.

$$W = W + \alpha \quad (3)$$
Algorithm 1 Simulated Annealing Algorithm

**Input:** $W_0, T, \alpha, k$

**Output:** An optimal keepout margin width;

1. Set the width of initial keepout margins to $W_0$;
2. while $t < T$ do
3. Refine placement by adding keepout margins in size of $W$;
4. if Overflow decreases then
5. Save placement;
6. Adjust keepout margins (3);
7. else
8. calculate acceptance probability $P$ (4);
9. if $P > r$ then
10. Adjust keepout margins (3);
11. else
12. Determine the optimal placement with lowest overflow;
13. Break;
14. end if
15. end if
16. Increase $t$ by 1;
17. end while

$$P = \exp \left( \frac{R}{k} \right)$$ (4)

2. Ant colony optimization mode: Simulated annealing algorithm optimization process is a serial way which is time-consuming. Ant colony optimization algorithm provides a parallel way to search the optimal solution, since the transfer of each ant is independent. In addition, it is characterized by its positive feedback as well as good robustness. Hence, SA is replaced with ACO to improve search mechanism.

According to the congestion elimination requirements, the evaluation function, transferring strategies and pheromone volatilization process of ACO algorithm are modified, for the sake of obtaining the global optimal solution with a smaller ant colony size and iteration.

The overall flow of ACO algorithm is summarized in Algorithm 2. First of all, the critical parameters ant number $ant\_num$, iteration $T$, transfer acceptance probability $p_0$ and pheromone volatilization rate $\rho$ are set in advance which play a decisive role in optimizing quality.

In step 1, we initialize keepout margin width of each ant randomly by formula (5) in which $w(i, t)$ represents the width of $i^{th}$ ant at the $t^{th}$ transfer iteration, and $w(i, 0)$ is the initial keepout margin width of $i^{th}$ ant. $lb$ and $ub$ are the lower limit and upper limit of the width. $lb = 0.01, ub = row\_height$. $r$ is a random number between 0 and 1.

$$w(i, 0) = lb + (ub - lb) \cdot r \quad i \in (0, ant\_num - 1)$$ (5)

In step 2–4, $r(i, t)$ is the pheromone concentration of $i^{th}$ ant at the $t^{th}$ transfer iteration, and $r(i, 0)$ is the initial pheromone of $i^{th}$ ant which is initialized to 0.

Step 5–23 is the core cycle for ant transfer. In step 6, we search the ant $i$ with the maximum pheromone concentration $r_{max}$. Based on the results, the formula (6) is adopted to evaluate each ant $i$ in step 8. The higher the pheromone $r(i, t)$, the lower the evaluated value $G(i, t)$.

$$G(i, t) = 1 - \frac{r(i, t)}{r_{max}(t)}$$ (6)

In step 9–14, the temporary change in the width is predicted by the formula (7) based on the relationship between evaluated results $G(i, t)$ and transfer acceptance probability $p_0$. In this formula, $w_p(i, t)$ and $w(i, t)$ denote the predicted results and current results of the width separately. $w_{init}(t)$ represents the width that can achieve lowest overflow at $t^{th}$ iteration.

$$w_p(i, t) = \begin{cases} w(i, t) + \frac{w_{init}(t) - w(i, t)}{r + 2} & p_0 \geq G(i, t) \\ w(i, t) + (r - 0.5) \cdot (ub - lb) & p_0 < G(i, t) \end{cases}$$ (7)

Finally, formula (8) is used to decide whether the width in next iteration is updated to the predicted width $w_p$ in step 15–20. Function $O(w)$ represents the overflow value achieved by keepout margin width $w$. If the predicted width gets worse overflow value, it will be abandoned. After the renewal process, in step 21 the pheromone volatilization is executed by the formula (9) where the volatilization $r(i, t + 1)$ for next iteration is obtained.

$$w(i, t + 1) = \begin{cases} w(i, t) & O(w_p) \geq O(w) \\ w_p(i, t) & O(w_p) < O(w) \end{cases}$$ (8)

$$r(i, t + 1) = (1 - \rho) \cdot r(i, t) + \frac{1}{O(w(i, t + 1))}$$ (9)

3. Experiment results

3.1 Experiment setup

We build an EDA verification experiment platform to verify and analyze the proposed algorithms. This experiment is composed of three procedures. The first is placement and congestion elimination, which is carried out on a Linux machine with Intel(R) Xeon 2.40 GHz and 64 GB memory. At this stage, normal placement operation is implemented on the post-synthesis netlist. And then, exert fast global routing
Table I Placement information.

|                | RISC-180nm | AE-55nm |
|----------------|------------|---------|
| Chip Size (um²)| 1153*1149  | 171*171 |
| Cell Area (um²)| 1876831    | 20123   |
| Cell Number    | 41,690     | 7,247   |

Fig. 3 Congestion reduction results-RISC-180nm.

Fig. 4 Congestion reduction results-AE-55nm.

procedure to obtain overflow value. Finally, according to the generated overflow value fed back from the second procedure, the proposed congestion elimination method adjusts the chip placement parameters in real time and produce the post-placement back to first procedure for congestion elimination. The proposed method is programmed by TCL and Python language.

A RISC processor core synthesized with SMIC 180 nm technology library and arithmetic encoder (AE) synthesized with SMIC 55nm technology library are selected as the experimental subjects to prove the feasibility of our proposed method. The placement information is listed in Table I.

3.2 Congestion removal results
In this section, we initially compare the congestion optimization effects of our proposed method with that of Synopsys’s IC Compiler (ICC), and then compare with paper [30] to analyze the pros and cons. The global routing congestion maps are depicted in Figure 3 and Figure 4, the severity of congestion determines the highlighting color, serious congestion is hotter.

In Figure 3 (a), there are some red blocks that lack routing resources in the lower middle part of the initial placement. After being optimized by EDA tools, severe congestion has been alleviated to some extent, however, there are a few new red blocks in the upper part. By using our proposed technique, in Figure 3 (c), the most congested regions in the lower middle part are alleviated without causing new congestion. In Figure 3 (d), compared with the initial design, the congestion has dropped significantly.

The other placement with tighter constraints is shown in Figure 4 (a), filled with thousands of red blocks. By ICC’s congestion optimization, in Figure 4 (b) congestion in the upper left corner is eased, but it is still spreading throughout the whole chip. After being optimized by our proposed technique, congestion distribution becomes more sparse. Although there is still many congested regions, the proportion of blue blocks has increased significantly compared with (b), indicating the overall congestion has been reduced.

More detailed data comparison is depicted in Figure 5, in which the optimization rate represents the drop rate relative to the initial placement, including design rule violations (DRV), short, and total wire length (TWL), the higher the value, the better the optimization effect. The experimental results show clearly that our proposed method can decrease these metrics more significantly and can outperform the ICC’s congestion optimization strategies. The ACO algorithm has advantages over SA algorithm in most cases. TWL is decreased due to fewer detoured long wires after congestion reduction.

It is worth noting that there are still several violations after optimization, this is because in order to prove the superiority of our method, we generally choose the design with severe congestion as our experimental subject. If the congestion is light, both Synopsys ICC and our technique can completely eliminate it, which would make no sense for comparison.

In [30], the TWL gets worse than the initial placement. Although short number is decreased from initial placement, the optimization effect is not as good as ICC. This is because it selects complex cells with 4 or more pins as optimized objects. The large number of selected cells will introduce many keepout margins in the chip, which instead causes new local congestion. And the width of keepout margins is an unchanged factor, resulting in insufficient optimization effects. However, our method can reduce both short and TWL significantly, and outperform ICC, this is because we only optimize the cells with the highest pin number and total pin number, which reduces the negative impact of keepout margins on the chip, and use two heuristic algorithms to calculate the most suitable keepout margin width. But our approach is more time-consuming due to multiple iterations.

3.3 Algorithm analysis
In this section, different factors are compared and analyzed under the design of AE-55nm, and short drop rate is selected as the evaluation index.

(1) Number of optimized regions: During the optimization process, if the number of optimized congested regions is small, the algorithm can not reach the maximum efficiency. Conversely, if the number is too large, not only waste time, but also induce a large amount of keepout margins, which will occupy additionalchip’s area and cause new congestion.

Demonstrated in Figure 6, the short drop rate is high when the number of optimized regions is in the range of
So, on the basis of a trade-off between the runtime and optimization effect, 3–4 regions are the best.

(2) Initial keepout width of SA: The initial keepout width is a key parameter in terms of the searching mechanism of SA. An appropriate width is able to improve both convergence and effectiveness of the algorithm. Demonstrated in Figure 7 where we choose 1, 1/2, 1/3, 1/4 times the width of one row height as the initial keepout width to analyze its impact on the algorithm. In the 10 iterations, 1/2 and 1/3 times can converge swiftly to the best result. 1 times can reach a best optimization effect with a long iteration, since a high initial width can provide more searching space for algorithm, it has more chance to obtain optimal value. It’s worth noting that only the 1/3 times can surpass ICC in the shortest time.

In summary, we suggest that if the designers intend to obtain a better optimization results as soon as possible, the initial keepout width should be 1/3 times as large as one row height. All curves are similar, and 1/2 times can reach a highest rate. Moreover compared with Figure 7, most can achieve a higher short drop rate after only 2 iterations, demonstrating that the efficiency of ACO algorithm.

(4) Ant number for ACO: Parameters such as ant number also play significant roles for the optimization effect. As depicted in Figure 9, as the number of ant increases, the short drop rate fluctuates due to the randomness of the algorithm. When the number exceeds 6, it tends to reach a plateau. Actually, the number of 3–5 ants can already achieve an acceptable optimization effect in a shorter runtime.

4. Conclusion

In this letter, we propose a local congestion elimination technique compatible with available commercial P&R EDA tools. The proposed method searches the optimal keepout margins being added around the highest pin cells in specific congestion regions using the heuristic algorithms. Compared with the results optimized by Synopsys ICC, the proposed technique can reduce the DRV, shorts and TWL effectively in different design of different process nodes.

Acknowledgments

The work is supported by National Natural Science Foundation of China (61504110, 61531016), Sichuan Science and Technology Program (2019YFG0092), National Natural Science Foundation of China (6177140, 61804128).

References

[1] D.Z. Pan, et al.: “Design for manufacturing with emerging nanolithography,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 32 (2013) 1453 (DOI: 10.1109/TCAD.2013.2276751).

[2] A. Jahanian and M.S. Zamani: “Metro-on-chip: an efficient physical design technique for congestion reduction,” IEICE Electron. Express 4 (2007) 510 (DOI: 10.1587/elex.4.510).

[3] C.J. Alpert, et al.: “What makes a design difficult to route,” Proc. ISPD (2010) 7 (DOI: 10.1145/1735023.1735028).

[4] Z. Li, et al.: “Guiding a physical design closure system to produce easier-to-route designs with more predictable timing,” Proc. DAC (2012) 465 (DOI: 10.1145/2228360.2228442).

[5] W.-T.J. Chan, et al.: “BEOL stack-aware routability prediction from placement using data mining techniques,” Proc. ICCD (2016) 41 (DOI: 10.1109/ICCD.2016.7753259).

[6] C.-H. Hsu, et al.: “Multi-layer global routing considering via and wire capacities,” Proc. ICCAD (2008) 350 (DOI: 10.1109/ICCAD.2008.4681597).

[7] Y. Wei, et al.: “Techniques for scalable and effective routability evaluation,” ACM Trans. Design Autom. Electr. Syst. 19 (2014) 17 (DOI: 10.1145/2566663).
[8] H. Shojaei, et al.: “Planning for local net congestion in global routing,” Proc. ISPD (2013) 85 (DOI: 10.1145/2451916.2451940).

[9] D. Shi and A. Davoodi: “TraPL: track planning of local congestion for global routing,” Proc. DAC (2017) 19 (DOI: 10.1145/3061639.3062335).

[10] J. Lou, et al.: “Estimating routing congestion using probabilistic analysis,” Proc. ISPD (2001) 112 (DOI: 10.1145/369691.369749).

[11] J. Westra, et al.: “Probabilistic congestion prediction,” Proc. ISPD (2004) 204 (DOI: 10.1145/981066.981110).

[12] C. Wang, et al.: “DLAU: a scalable deep learning accelerator unit on FPGA,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 36 (2017) 513 (DOI: 10.1109/TCAD.2016.2587683).

[13] Z. Qi, et al.: “Accurate prediction of detailed routing congestion using supervised data learning,” Proc. ICCAD (2014) 97 (DOI: 10.1109/ICCAD.2014.6974668).

[14] W.-J. Chan, et al.: “Routability optimization for industrial designs at sub-14nm process nodes using machine learning,” Proc. ISPD (2017) 15 (DOI: 10.1145/3036669.3036681).

[15] Q. Zhou, et al.: “An accurate detailed routing routability prediction model in placement,” Proc. ASQED (2015) 119 (DOI: 10.1109/ASQED.2015.7274019).

[16] A.F. Tabrizi, et al.: “A machine learning framework to identify detailed routing short violations from a placed netlist,” Proc. DAC (2018) 1 (DOI: 10.1109/DAC.2018.8465835).

[17] J.A. Roy and I.L. Markov: “High-performance routing at the nanometer scale,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 27 (2008) 1066 (DOI: 10.1109/TCAD.2008.923255).

[18] M.D. Moffitt: “MaizeRouter: engineering an effective global router,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 27 (2008) 2017 (DOI: 10.1109/TCAD.2008.2006082).

[19] M.M. Ozdal and M.D.F. Wong: “Archer: a history-driven global routing algorithm,” Proc. ICCAD (2007) 488 (DOI: 10.1109/ICCAD.2007.4397312).

[20] Y.-J. Chang, et al.: “NTHU-Route 2.0: a fast and stable global router,” Proc. ICCAD (2008) 338 (DOI: 10.1109/ICCAD.2008.4681595).

[21] H.-Y. Chen, et al.: “High-performance global routing with fast overflow reduction,” Proc. ASP-DAC (2009) 582 (DOI: 10.1109/ASPDAC.2009.4796543).

[22] Y. Xu, et al.: “FastRoute 4.0: global router with efficient via minimization,” Proc. ASP-DAC (2009) 576 (DOI: 10.1109/ASPDAC.2009.4796542).

[23] K.-R. Dai, et al.: “NCTU-GR: efficient simulated evolution-based rerouting and congestion-relaxed layer assignment on 3-D global routing,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 20 (2012) 459 (DOI: 10.1109/TVLSI.2012.2234489).

[24] T.-H. Wu, et al.: “A parallel integer programming approach to global routing,” Proc. DAC (2010) 194 (DOI: 10.1109/1837274.1837323).

[25] W.-H. Liu, et al.: “Multi-threaded collision-aware global routing with bounded-length maze routing,” Proc. DAC (2010) 200 (DOI: 10.1109/1837274.1837324).

[26] Y. Han, et al.: “Exploring high-throughput computing paradigm for global routing,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 22 (2014) 155 (DOI: 10.1109/TVLSI.2012.2234489).

[27] W.-H. Liu, et al.: “A fast maze-free routing congestion estimator with hybrid unilateral monotonic routing,” Proc. ICCAD (2012) 713 (DOI: https://doi.org/10.1109/2429384.2429539).

[28] A. Farkish and A. Jahanian: “Parallelizing the FPGA global routing algorithm on multi-core systems without quality degradation,” IEICE Electron. Express 8 (2011) 2061 (DOI: 10.1587/elex.8.2061).

[29] Z. Qi, et al.: “VFGR: a very fast parallel global router with accurate congestion modeling,” Proc. ASP-DAC (2014) 525 (DOI: 10.1109/ASPDAC.2014.6742945).

[30] F. Bodine: “Reducing cell placement congestion using targeted pattern halos,” SNUG Proc. PD (2014) 1.