1-D GDR Aware Cell Generation via P/N Bi-Partition

Yao-Lin Chang, Chien-Hung Lin, Wei-Tung Chao, Hung-Ming Chen
Institute of Electronics, National Chiao Tung University, Hsinchu, Taiwan
Email: grlu717.ee02g@g2.nctu.edu.tw, lingege32.ee06g@nctu.edu.tw,
ddcfei.ee07g@nctu.edu.tw, hmchen@mail.nctu.edu.tw

Abstract— As the complexity of layout design grows, layout generation problem has been more challenging. This work features the bi-partition tree and the selective stage for cell generation. With this bi-partition tree, we speed up the layout generation flow and minimize the wire length. With objective functions in the placement selection stage and the routing stage, a lithography-friendly layout with low congestion, minimized area and high performance can be accomplished.

I. Introduction

Automated cell layout generation has been discussed for decades since [1] solved it with Euler Path. However, there is still no general solution for all kinds of layout design because of the difficulty and rapid development in VLSI field. As technology node scales with highly complex design rules, it is necessary to move on to 1-D GDR (Gridded Design Rule) layout style [2] in more advanced technology. 1-D GDR is better than traditional 2-D CDR (Complex Design Rule) in area, power dissipation and optical proximity correction, as shown in [2]. As demonstrated in [3], 1-D GDR is better generated by placement algorithms based on P/N stacks selection, because all the layers in this style have only one direction. The placement algorithm based on P/N stacks selection [4] [5] separates transistors by their types at first. Then, P and N transistors do permutation individually, and we can finally merge P and N stacks together to find the best solution.

II. Our Design Planning

A layout generation algorithm is proposed in this section. In our algorithm, we first input a netlist. Then we partition transistors into a bi-partition tree. After the tree is constructed, we place transistors in the leaves. With placements in leaves, we merge them into the final placement. Finally, we perform symbolic routing to determine if the placement is valid or not. If it is invalid, we add one unit width at placement stage and redo the leaf placement and symbolic routing. Otherwise, we perform post simulation in the output.

A. *Bi-Partition for P&N*

In our algorithm, we bi-partition a set of transistors into two subsets as a tree until the number of transistors in leaves is less than 20. To choose the best bi-partition result, we define $DIFF(P)$ as the difference of the number of transistors in subsets, $Cut(P)$ as the number of the same nets cut, and the additional width estimation $W(P) = 2 \cdot max\{0, n_{odd} - 2\}$ with the bi-partition result P. n_{odd} is the number of nets with the same odd number of terminals. For example, we have a subset with nets 1,1,1,2,2,4,7. Then, $W(P)$ is 2 with n_{odd} is 2 for net 4,7 with one terminal.

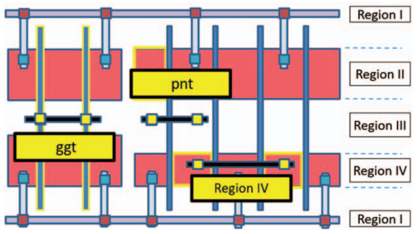

Fig. 1. Routing Region. Region I,V are region for power, Region II is P region, region IV is N region, and Region III is middle region.

Lemma II.1 *We have the same wire length if Cut(P) is less or equal to 2.*

Based on Lemma.II.1, given a bi-partition result P, we accept it only when $Cut(P)$ is the minimum number or less than three. Then when the $Cut(P)$ is qualified, we have W(P) as smaller as possible. Last, we update our solution if the previous cost is acceptable and we have smaller $DIFF(P)$.

B. *Find the Best Placement of P/N stacks*

In this stage, we define ST_p as P-type transistors stack, ST_n as N-type transistors, and five regions as shown in Fig. 1.

Definition II.1 *For horizontal tracks in Region III, we have two kinds of tracks in a hyperedge $e \in E$ as shown in Fig.1:*

ggt_e: Longest $track_e$ from gate to gate.

pnt_e: Shortest $track_e$ from ST_p to ST_n.

With ggt_e and pnt_e, we have x_{r1}, x_{l1}, x_{r2} and x_{l2}, and we define:

$$track_e^{III}: (min(x_{l1}, x_{l2}), max(x_{r1}, x_{r2}))$$

As shown in Definition II.1, after ggt and pnt are assigned, we merge them to be $track_e$ in Region III. After the tracks in Region III are decided, we assign tracks of nets only on Region II or Region IV by simply connecting from the leftmost position to the rightmost. Then we define $WL(P)$ as total wire length of nets. To minimize the congestions of the placement, we estimate the minimum height of the tracks assignment $H(PC)$ in Region II, III and IV. Therefore, we find the solution that transistors in the same net are closer and are minimized the congestions.

We still need to consider the I/O nets from outside the cell and cut nets because of the bi-partition. To do so, we define a cost function to determine how close they are to the border. Given a hyperedge $e \in E$, we have two border l and r, which are respectively represented the leftmost and rightmost border. Take the leftmost border l as an example, we define $\mathbf{dis_e^l}$ as the distance from a closest

TABLE I
PERFORMANCE COMPARISON TO STANDARD CELL LIBRARY.

Cell Name	Standard Cell Library			Our work				Compare		
	Area (μm^2)	T_r/T_f (ps)	Power (μW)	Area (μm^2)	T_r/T_f (ps)	Power (μW)	Time (s)	Area (%)	T_r/T_f (%)	Power (%)
AND4	1.827	10.4/3.6	1.82	1.6905	10.2/3.71	1.86	0	-7.47	-1.92/+3.05	+2.2
OAI222d2	1.974	7.38/12.9	0.634	1.827	7.31/12.1	0.649	0	-7.447	-0.949/-6.2	+2.37
AOI222d2	1.974	12.5/2.61	0.796	1.827	11.4/3.1	0.859	0	-7.447	-8.8/+18.77	+7.915
MAOI222	1.386	5.46/5.38	1.83	1.281	5.3/5.67	1.81	0	-7.576	-2.93/+5.4	-1.09
OR4	0.798	2.42/7.42	1.72	0.735	2.53/7.37	1.75	0	-7.895	+4.54/-0.67385	+1.744
XOR4	3.454	6.04/11	3.32	4.598	5.95/10.5	3.52	0	+33.121	-1.49/-4.545	+6.024
Decoder	3.591	9.74/8.04	11.9	2.97	9.24/8.13	11.6	0	-17.29	-5.133/+1.12	-2.52
Latch	2.268	5.17/4.44	3.28	2.91	5.16/4.53	3.45	0	+28.3	-0.95/-6.2	+2.37
XNR4	3	7.42/7.02	3.23	3.192	6.52/6.43	3.45	0	+6.4	-12.13/-8.405	+6.81
DFF	2.268	5.42/6.13	4.74	2.919	5.69/6.68	5.18	0	+28.703	+4.982/+8.97	+9.28
Avg.	-	-	-	-	-	-	-	+4.14	-2.578/+1.128	+3.51

position of terminal e to the leftmost side of the layout design. Then, we have two cost functions for I/O_edge defined in netlist file and Cut_edge defined in bi-partition, they are respectively regarded as the distance to the leftmost border and the rightmost border.

Definition II.2 *Cut edge distance and I/O edge distance. Given a set of cut nets C and a set of I/O nets I, we define:*

$$CutDis(PC) = \sum_{c \in C} dis_c^l$$

$$IODis(PC) = \sum_{i \in I} dis_i^r$$

As shown in Definition II.2, we can shorten the distance between the transistors that are divided into different groups by the bi-partition process with $\mathbf{CutDis(P)}$ and pull transistors with I/O nets to the border for inter-cell routing with $\mathbf{IODis(P)}$. After defining the objective function, we introduce how to select our best placement based on the definition above. To find the best solution, we choose among all the possible combination of P/N stacks with the objective functions.

In the selection stage, we accept the solution that have better cost in objective function with high priority first. For example, if a placement has better cost in the highest priority objective function. First we find the placement with the smallest gate matching numbers $\mathbf{GM(PC)}$. With the same $\mathbf{GM(PC)}$, we choose a placement with the minimum $\mathbf{H(P)}$. Second, with the same cost mentioned above, to make shorter wire length in nets cut in bi-partition process, we choose the smallest $\mathbf{CutDis(P)}$.

To have closer diffusion terminals and gate terminals, we want to find the minimum $\mathbf{WL(P)}$. After the above objective functions, positions of transistors are roughly determined. For the last priority, $\mathbf{IODis(P)}$ will be considered since the IO net is considered for inter-cell routing process. Finally, with stacks enumeration algorithm, we can find the solution with the minimum area. Moreover, based on Definition II.1, we have high performance and routability by minimizing capacitance and wire length.

C. Lithography Aware Symbolic Routing

To handle multiple nets routing problem, we apply the maximum independent set (MIS) algorithm. However, it is not appropriate to apply greedy MIS algorithm listed in [6] because the solution is not valid for a real layout sometimes. To make sure the solution available to be assigned onto the layout canvas in commercial tools, we enumerate all possible routing and select the solutions with the minimal area that satisfy stage-like line end gap problem demonstrated in [7].

III. Experimental Results

In this section, we present experimental results of the proposed algorithm. We test our algorithm based on 28nm process 12 tracks SVT Standard Cell Library and PDK in TT mode. To demonstrate the performance of our final layout, we compare our result with the leading foundry 28nm Standard Cell Library layout with area, power dissipation and T_r/T_f in Table I. In these values, area is calculated by the width of a layout multiplying the height of a layout. Power dissipation is calculated by the average current I multiplying the power voltage V. T_r/T_f is the skew of the layout.

IV. Conclusion

In this short paper, we present an efficient and deterministic algorithm to solve 1-D layout generation problem with high printability. In the algorithm, we speed up the process by applying bi-partition tree and minimize the congestion of the placement result with proper objective functions. Eventually, we optimize the layout performance in area, timing and power dissipation and eliminate stage-like line end gaps at the same time. Results show that we can handle standard cells in only few seconds and the post-simulation result is considerably good in area, timing and power dissipation.

References

[1] T. Uehara and W. M. vanCleemput, "Optimal layout of cmos functional arrays," *IEEE Trans. CAD*, vol. 30, no. 5, pp. 305–319, 1981.

[2] M. Smayling, "Gridded design rules: 1d approach enables scaling of cmos logic," *Nanochip technology*, vol. 6, no. 2, pp. 33–37, 2008.

[3] P.-H. Wu, M. P.-H. Lin, T.-C. Chen, T.-Y. Ho, Y.-C. Chen, S.-R. Siao, , and S.-H. Lin, "1-d cell generation with printability enhancement," *IEEE Trans. CAD*, vol. 32, no. 3, pp. 419–432, 2013.

[4] A. Ziesemer and C. Lazzar, "Transistor level automatic layout generator for non-complementary cmos cells," in *IFIP International Conference on Very Large Scale Integration*, 2007, pp. 116–121.

[5] A. Ziesemer, R. Reis, M. T. Moreira, M. E. Arendt, and N. L. V. Calazans, "Automatic layout synthesis with astran applied to asynchronous cells," in *Latin American Symposium on Circuits and Systems*, 2014, pp. 1–4.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2009.

[7] H. Zhang, M. D. Wong, and K.-Y. Chao, "On process-aware 1-d standard cell design," in *Proc. DAC*, 2010, pp. 838–842.