Practical ILP-based Routing of Standard Cells

Hsueh-Ju Lu, En-Jang Jang, Ang Lu, Yu Ting Zhang, Yu-He Chang, Chi-Hung Lin, Rung-Bin Lin
Department of Computer Science and Engineering
Yuan Ze University
Taoyuan, Taiwan

***Abstract*—This paper proposes a two-stage transistor routing approach that synergizes the merits of channel routing and integer linear programming for CMOS standard cells. It can route 185 cells in 611 seconds. About 21% of cells obtained by our approach have smaller wire length than their handcrafted counterparts. Only 11% of cells use more vias than their handcrafted counterparts. Our router completes routing of many cells that cannot be routed by an industrial one.**

Keywords—Routing; standard cell; transistor; integer linear programming

I. INTRODUCTION

As process technology advances rapidly towards more layout regularity for better manufacturing yield [1,2], automatic layout generation that can best take advantage of layout regularity becomes a viable approach to physical synthesis of standard cells. Physical synthesis of standard cells has two major tasks: transistor placement and routing. Transistor placement [3] though is difficult, it is less dependent on process technology. On the contrary, transistor routing highly dependent on process technology is more difficult. It requires achieving highly optimized result while still efficiently dealing with complicate layout design rules. In early days when layout design rules were not so complicate, transistor routing could be done using left-edge algorithm [4], a channel routing algorithm [5,6], an area router assisted by a compactor [7], pattern routing [8], Boolean satisfiability(SAT) approach [9-12], etc. The work in [12] gives an excellent review of transistor routing methods. Especially, the SAT approach in [12] can handle layout design rules robustly. It is accompanied by an integer linear programming (ILP) approach for wire length and via minimization. The ILP solution is highly dependent on the given SAT solution. Though this method [12] can obtain smaller wire length, it produces an excessive number of vias.

In this article we propose a two-stage transistor routing approach for CMOS standard cells. The first stage determines the horizontal span of a net. The second stage employs ILP to assign horizontal wire segments to routing tracks while taking all the layout design rules into account. Our approach simplifies the placement of multiple access pins for an input signal. The experimental results show that our approach can route 185 cells in 611 seconds on a 32-core machine. About 21% of cells have smaller wire length than their handcrafted counterparts. The overall wire length is on average 1.26% larger than that obtained by handcrafting. Moreover, 89% of cells obtained by our approach do not use more vias than their handcrafted counterparts. An automatic router embedded in an industrial layout editor is also employed to route the same 185 cells. However, it can only route 18 cells successfully. We also port our router to work for a 32nm [13] and a 45nm process technology [14].

The rest of the paper is organized as follows. Section II presents our problem formulation. Section III details our routing methodology. Section IV presents some experimental results. Section V draws conclusions.

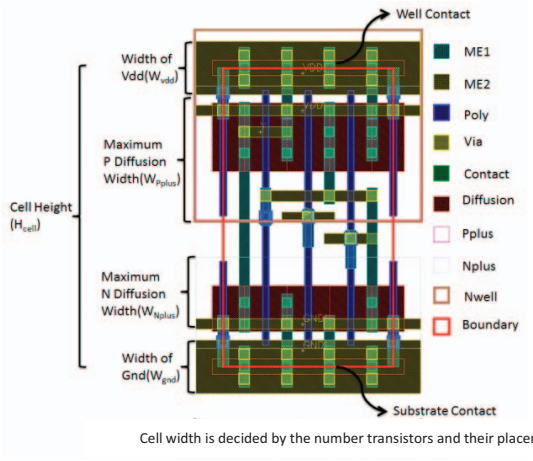
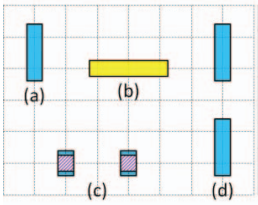


Fig. 1. Layout template.

TABLE I. LAYOUT RULES OBEYED BY OUR ROUTER.



Rule	Description
(a)	Length of an ME1 wire segment is at least 2 grids
(b)	Length of an ME2 wire segment is at least 3 grids
(c)	Two vias must be separated by 1 grid
(d)	Two line ends of ME1 wire segments must be separated by 1 grid

II. PROBLEM FORMULATION

In this work transistor routing is performed to complete layout design of P- and N-transistors placed on a two-row fabric. The transistor placement is embedded into a layout template as shown in Fig. 1. The template based on a 90nm industrial process technology consists of some routed wires that implement a 2-input AND gate. Poly, ME1, and ME2 layers each have a fixed pitch. A wire on any of these layers cannot bend. ME1 wires are laid vertically and ME2 wires are laid horizontally only (called 1-D layer constraints). The template's height is 3.92μm and provides 9 ME2 tracks. If the sources/drains of two adjacent transistors are connected to different signals, the two transistors should be separated by an isolation transistor. An isolation transistor is also deployed on the left and right boundary respectively. The boundary is designed to facilitate cell and row abutment. The template has a thick and a thin ME2 power rail. The thick rail provides current to functional transistors. The thin rail provides a steady voltage to an isolation transistor's poly gate to turn off the transistor. We will place as many connections as possible between these two rails. Adopting a fixed pitch respectively on poly, ME1, and ME2 layers creates a gridded routing fabric which simplifies layout design rules and hence reduces routing complexity. In this work, our router obeys the layout rules given in Table I.

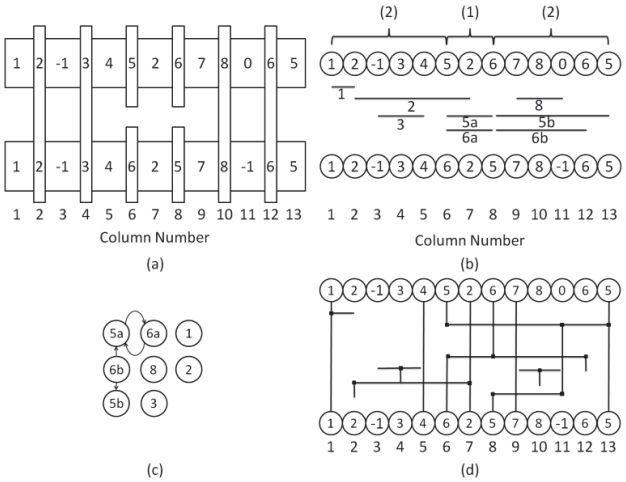


Fig. 2 (a). Transistors placement, (b). Horizontal wire segments, (c).Vertical constraint graph, (d) Routing result.

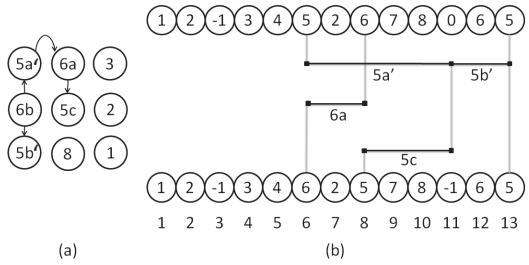


Fig. 3. (a) Updating VCG, (b) regenerating horizontal wire segments due to doglegging performed on 5a in Fig. 2(b).

Exploiting the layout template in Fig. 1 translates the transistor routing problem into a channel-like one as shown in Fig. 2(a). The sources/drains and gates of transistors are labeled with numbers. A "0" denotes the underlying column has no signal connection. A "-1" denotes that a terminal connects to a power or ground rail. A number greater than "0" denotes a terminal of a net. Terminals labeled with the same number greater than zero should be connected.

III. Routing Methodology

We take a two-stage approach to this routing problem. The first stage determines the horizontal span of a net. It uses a vertical constraint graph (VCG) to determine which wire segment should be broken for doglegging and where a dogleg should be located. Fig. 2(b) shows the horizontal wire segments of the underlying problem. This stage also sets up access pins (short wires on ME2) for each input signal. The second stage uses ILP to assign horizontal wire segments to routing tracks while taking all layout design rules into account. After this, we add vertical wire segments to complete routing as shown in Fig. 2(d). We further reduce ME1(vertical) wire length by moving the wire ends on the top downward and those at the bottom upward. The two stages are presented below.

A. Generating Horizontal Wire Segments

Generating horizontal wire segments is complicated by doglegging done to break a cycle or an excessively long path on a VCG. The questions are "which wire segment will be doglegged?" and "which column to place a dogleg?" Below is our approach.

- Determining a column for doglegging to break a cycle

We can break a cycle in a VCG by doglegging the wire segments that form the cycle. As shown in Fig. 2(c), wire segments 5a and 6a form a cycle. Hence, we check the region denoted by (1) in Fig. 2(b) to find a column for doglegging either wire segment 5a or 6a. Doglegging at this region will not increase horizontal wire length. We first select a column which has a label -1 or 0 both on the top and at the bottom. A dogleg placed at this column does not create new vertical constraints but will increase wiring density at this column. If the above selection is not possible, we look for a column which has a label -1 or 0 either on the top or at the bottom. Inserting a dogleg into this column will incur a new vertical constraint and increase wiring density at this column. If this is not possible either, we continue to look for a poly column where the two transistors have the same gate signal, i.e., the poly wire is not broken. The dogleg will be laid over the poly wire. Note that, based on the underlying layout design rules, the length of a wire segment selected for generating a dogleg in region (1) should not be smaller than five columns. Note that both wire segments 5a and 6a in Fig. 2 have a length less than five. Hence, we cannot dogleg them.

We proceed to check the columns in region (2) if no column in region (1) is found acceptable. Similarly, we select a column that will minimize horizontal wire length increase. As shown in Fig. 3, column 11 in region (2) is selected for doglegging with a minimum wire length increase of 3 columns. For ILP formulation introduced later, the set of wire segments such as {5a, 5b} should be replaced by a new set of wire segments {5a', 5b', 5c} as shown in Fig. 3. The VCG must be updated accordingly. Note that column 2 can also be selected, but it causes a larger wire length increase. If we can't find any column for doglegging, we will add a pair of isolation transistors at a place without increasing wire length and channel density.

To obtain a best doglegging solution, we try to create a dogleg at *each node* (i.e., each wire) of a cycle in a VCG. For *each node*, we use the method depicted above to find all the possible columns for doglegging. We then use left-edge algorithm to evaluate the number of routing tracks needed for each dogleg solution. After this is done for all the nodes of a cycle, we adopt the solution that requires a minimum number of routing tracks.

- Determining columns for doglegging to break a longest path

The minimum number of tracks required for routing is determined either by channel density or by the longest path of the underlying VCG. Due to side-by-side via rule (c) as shown in Table I, the number of routing tracks needed for a path on the VCG is twice its length minus one. Certainly, a solution does not exist if channel density exceeds the number of available routing tracks. Nevertheless, for the case of the longest path exceeding a predefined limit, we can obtain a solution by doglegging a node on the path. We seek to dogleg a minimum number of nodes. Let $v_1, v_2, \dots, v_y$ denote the longest path of length y and R denote the available routing tracks. Considering the side-by-side via rule (c), i.e., two adjacent vias at least separated by one track as shown in Fig. 4, we should have $y \le \lfloor (R+1)/2 \rfloor \equiv D$ in order to obtain a feasible solution. Then, the nodes selected for doglegging will be $v_D, v_{2D-1}, v_{3D-2}, \dots, v_{rD-r+1}$ where $rD - r + 1 < y \le (r+1)D - r$. As shown in Fig. 4(a), the longest path length is 4, i.e., $y = 4$. Hence, we need at least 7 tracks for routing. Now, supposed we have only three tracks(R=3) available. Since $D = 2$, we will make a dogleg at nodes v_2 and v_3 respectively. Fig. 4(b) shows the result of doglegging.

B. ILP Formulation for Placing Horizontal Wires

After all the horizontal wire segments are generated, each should be placed at a track such that no vertical constraints are violated. We approach this problem by formulating it as an ILP model. All related

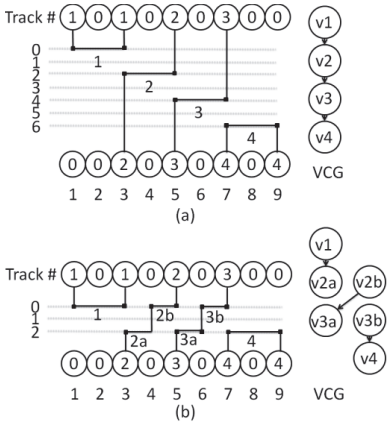


Fig. 4. Creating doglegs to break the longest path apart.

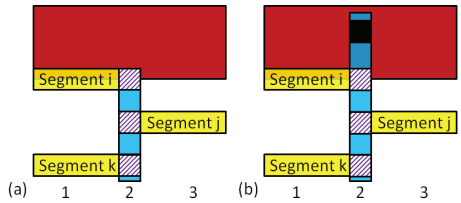


Fig. 5. ILP formulation for counting vias.

layout rules are considered. The objective function will minimize total vertical wire length, number of vias, and number of access pins on the same track. We install multiple access pins for each input signal and spread them to different horizontal tracks as much as possible. Note that the access pins for an output signal are created automatically on the ME2 wires that connect the output signal.

- Formulating vertical and horizontal constraints and via rules

Let T_i be an integer variable denoting the track assigned to horizontal wire segment i. If there is a vertical constraint from segment i to segment j , i.e., segment i should be placed at a track above j, then

$$T_j - T_i > 0 \tag{1}$$

Any two wire segments cannot be placed at the same track if there is a horizontal constraint between them. Hence, we have (2).

$$ABS_{ij} = |T_j - T_i| > 0 \tag{2}$$

Since any two vias joining ME1 and ME2 wires cannot be placed side by side if there is not a track between them, we have (3) for wire segments i and j if $|x_j - x_i| = 1$ or have (3') if $|x_j - x_i| = 0$ where x_i is the column position of a via on segment i and x_j is the column position of a via on segment j.

$$|T_j - T_i| \geq 1 \tag{3}$$

$$|T_j - T_i| \geq 2 \tag{3'}$$

- Calculating vertical wire length

Since the total length of horizontal wires is determined after the first stage is completed, we simply minimize the total vertical wire length. If a vertical wire segment f is connected to a P-transistor and also to horizontal wire segments $h_1, h_2, \ldots, h_k$ located at tracks $T_{h_1}, T_{h_2}, \ldots, T_{h_k}$ respectively, (4) calculates its length.

$$P_f = max\,(T_{h_1}, T_{h_2}, \ldots, T_{h_k}) \tag{4}$$

If f is connected to an N-transistor, (5) calculates its length.

$$N_f = 8 - min\,(T_{h_1}, T_{h_2}, \ldots, T_{h_k}) \tag{5}$$

If f is connected to neither a P-transistor nor an N-transistor, (6) calculates its length.

$$B_f = max(T_{h_1}, T_{h_2}, \ldots, T_{h_k}) - min(T_{h_1}, T_{h_2}, \ldots, T_{h_k}) \tag{6}$$

Then, total wire length is given in (7).

$$L_{vtotal} = \sum_{\forall f} P_f + \sum_{\forall f} N_f + \sum_{\forall f} B_f \tag{7}$$

- Counting vias

When more than one wire segment at different tracks need to be connected together, vias are used to connect them to a vertical ME1 wire g. To count vias, for wire segments i and j yet to be connected we check whether they are assigned to the same track, i.e., we compute $ABS_{ij} = |T_i - T_j|$. If $ABS_{ij} = 0$, segments i and j are on the same track. Basically, if all the horizontal wire segments of interest are on the same track and they don't need a vertical wire to connect them to diffusion, then no vias are needed or a via is needed otherwise (called a * case). Fig. 5(a) shows a case of the former whereas Fig. 5(b) shows a case of the latter. To come up with an exact number of vias, we count the occurrences of $ABS_{ij} > 0$. Let the counting result be denoted by M. Given two wires, if M equals 0, the number of vias V_g =0 (or V_g =1 for * case). If M equals 1, V_g =2. In case we are given three wire segments, V_g is 3, 2, 0 (1 for * case) respectively for M=3, 2, 0. The *if-then-else* statements in (8) map M to V_g for the case of five wire segments. Then, the number of vias is given in (9).

$$\text{if}(M > 2)\ then\ V_g \geq 2\ \text{else}\ V_g \geq 0\ (V_g \geq 1\ for * case)$$
$$\text{if}(M > 4)\ then\ V_g \geq 3\ \text{else}\ V_g \geq 2 \tag{8}$$
$$\text{if}(M > 5)\ then\ V_g \geq 4\ else\ V_g \geq 3$$

$$V_{total} = \sum_{\forall g} V_g \tag{9}$$

- Installing multiple access pins for an input signal

We consider installing multiple access pins for each input signal. An access pin is a short horizontal ME2 wire. Hence it should be included into our ILP formulation. To spread out access pins, we minimize the occurrences of placing them at the same track. Let $ABS_{ij} = |T_i - T_j|$ where T_i and T_j are track numbers assigned to access pins i and j. Let $b_{ij} = 1$ denote $ABS_{ij} = 0$ and $b_{ij} = 0$ denote $ABS_{ij} > 0$. The total number of access pins placed at the same track is given in (10). We would like to minimize C_{total}.

$$C_{total} = \sum_{\forall i,j} b_{ij} \tag{10}$$

- Complete ILP formulation

The complete ILP formulation is given in (11). The objective function minimizes total vertical wire length, total number of vias, and number of access pins on the same track. α, β, and γ are pre-specified weights.

$$Minimize: \alpha * L_{vtotal} + \beta * V_{total} + \gamma * C_{total}$$
$$Subject\ to\ all\ the\ constraints\ specified\ in\ (1)\sim(10). \tag{11}$$

IV. Experimental Results

Experiments are run on a 2.13GHz Intel Xeon (32 cores) with 192G bytes memory. In 611 seconds, our router completes routing of 185 logic cells based on the template in Fig. 1. *Cplex* [15] is employed to solve ILP models. The logic cells include commonly used ones with driving capabilities of 1X, 2X, and 4X. The driving capabilities of buffers and inverters are up to 32X. The transistor placement is obtained by the placer in [16]. We also manually perform routing of these 185 logic cells. The layout design is based on a 90nm industrial process technology whose layout rules are given in Table I. The values of α, β, and γ in (11) are set to 1, 3, and 4 respectively in our experiments.

TABLE II. RATIOS OF WIRE LENGTHS OF THE CELLS CREATED BY OUR ROUTER TO THAT OF THE CELLS CREATED BY HANDCRAFTING.

Ratio range	# of cells	Ratio range	# of cells
$-15\% < \text{ratio}_{rh} \leq -10\%$	4	$0\% < \text{ratio}_{rh} \leq 5\%$	103
$-10\% < \text{ratio}_{rh} \leq -5\%$	4	$5\% < \text{ratio}_{rh} \leq 10\%$	42
$-5\% < \text{ratio}_{rh} \leq 0\%$	31	$10\% < \text{ratio}_{rh} \leq 15\%$	1

$* W_r = wire\ legnth\ of\ cell\ created\ by\ our\ router$
$* W_h = wire\ length\ of\ cell\ created\ by\ handcrafting$
$*ratio_{rh} = \frac{W_r - W_h}{W_h}$

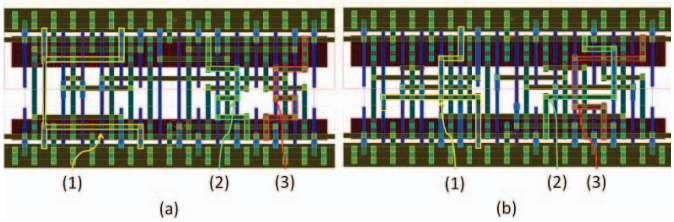


Fig 6. MUX4X2 created by (a) handcrafting and (b) our router.

TABLE III. DIFFERENCE IN NUMBER OF VIAS (A: OUR APPROACH; H: HANDCRAFTING).

Difference= A-H	# of cells	Difference= A-H	# of cells
-4	1	2	4
-3	1	3	0
-2	3	4	0
-1	20	5	0
0	140	6	1
1	15	7	0

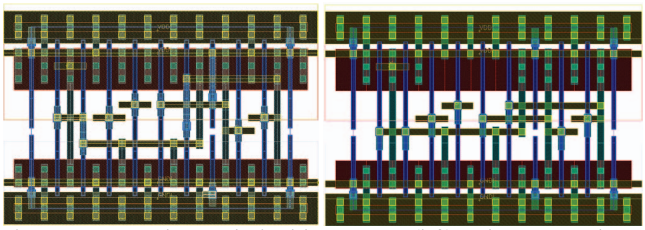


Fig. 7. OR4B2X2 layout obtained by *C-router* (left) and OR4B2X2 layout obtained by our router (right).

Table II is a comparison of wire length. About 21% of the cells (39 out of 185) obtained by our router have smaller wire length than their handcrafted counterparts. The total wire length achieved by our router is on average 1.26% larger than that obtained by handcrafting. Fig. 6 gives the routing results of MUX4X2 which has three cycles (1), (2) and (3) in its VCG. For destroying cycle (1), the handcrafted cell use a poly gate directly as a dogleg, but our router uses only ME1 and ME2 wires. However, for destroying cycles (2) and (3), the handcrafted cell uses more vias. This can also explain why the handcrafted cells may use more vias as shown in Table III. Table III shows that 13% of cells obtained by our router use fewer vias than their handcrafted counterparts and 11% of cells use more vias.

An automatic router embedded in an industrial layout editor is also architected to route the cells. We called this router *C-router*. *C-router* does not have an option to set up 1-D layer constraints. Instead, it uses weights to penalize wrong-way routing. In order to reduce wrong-way routing, we set wrong-way penalty to its maximum value. The results obtained from *C-router* are as follows. Sixty cells fail to complete routing, 107 cells contain wrong-way wires, and 18 cells have correct routing. Each of the 18 correct cells has worse wire length than its counterpart obtained by our router. The via counts are also worse than or equal to that obtained by our router. Fig. 7 shows such an example. The layout obtained by *C-router* also tends to have more detours.

Usually, a router must be modified to adapt to a new rule set for another process technology. Fortunately, the regularity of gridded routing fabrics eases the porting task of our router. We have ported our router to perform routing based on a 32nm process technology [13]. Another porting is for FreePDK45 technology [14]. Note that our router will not work for the cell design style where power and ground rails run directly over diffusions as shown in [8].

V. CONCLUSIONS

This paper proposes a two-stage transistor routing approach for CMOS standard cells. This two-stage approach synergizes the merits of channel routing and ILP approach. The segment-based ILP makes our router more scalable than a grid-based ILP. Our router achieves a routing quality comparable to the handcrafting approach and a quality much better than an automatic router embedded in an industrial layout editor. Our router can practically generate cell layouts which may be employed directly for production, benchmarking the handcrafted layouts, or making an early evaluation of design methodology and process technology.

REFERENCES

[1] V. V. Rovner, Circuit-Layout Co-optimization for Extremely Regular Design Fabrics in Nanoscale ICs, PhD Thesis, ECE, Carnegie Mellon University, 2011.

[2] M. P. Sole, Layout Regularity for Design and Manufacturability, Ph.D. Thesis, Universitat Polit_ecnica de Catalunya, 2012.

[3] C. Y. Hwang, Y. C. Hsieh, Y. L. Lin and Y. C. Hsu, "A fast transistor-chaining algorithm for CMOS cell layout." IEEE TCAD, no 7, pp. 781-786, July 1990.

[4] Y.C. Hsieh, C. Y. Hwang, Y. L. Lin, and Y. C. Hsu, "LiB: A CMOS cell compiler," IEEE TCAD Vol. 10. no. 8, pp. 994-1005, Aug. 1991.

[5] K. Tani, K. Izumi, M. Kashimura, T. Matsuda and T. Fujii,"Two-dimensional layout synthesis for large-scale CMOS circuits," in Proc. ICCAD, pp. 490-493, 1991.

[6] C.-L. Ong, J.-T. Li. and C.-Y. Lo, "GENAC: An automatic cell synthesis tool," in Proc. DAC, pp.239-244, 1989.

[7] M. Guruswamy, R. L. Maziasz, D. Dulitz, S. Raman V. Chiluvuri, A. Fernandez, and L. G. Jones, "Cellerity: a fully automatic layout synthesis system for standard cell libraries," in Proc. DAC, pp. 327–332, 1997.

[8] N. Ryzhenko and S. Burns, "Physical synthesis onto a layout fabric with regular diffusion and polysilicon geometries," in Proc. DAC, pp. 83-88, 2011.

[9] B. Taylor and L. Pileggi, "Exact combinatorial optimization methodsfor physical design of regular logic bricks," in Proc. DAC, pp. 344–349, 2007.

[10] W. Hung, X. Song, T. Kam, L. Cheng, and G. Yang, "Routability checking for three-dimensional architectures," IEEE TVSLI, vol. 12, no. 12, pp. 1371–1374, Dec. 2004.

[11] N. Ryzhenko and S. Burns, "Standard cell routing via Boolean satisfiability," in Proc. DAC, pp. 603–612, 2012.

[12] J. Cortadella, J. Petit, S. Gomez, and F. Moll, "A Boolean rule-based approach for manufacturability-aware cell routing," IEEE TCAD, Vol. 33, No. 3, pp. 409-422, March 2014.

[13] SAED32/28nm 1P9M 1.05V/1.8V/2.5V, Synopsys.2012.

[14] http://www.eda.ncsu.edu/wiki/FreePDK45:Contents

[15] http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

[16] A. Lu, H. J. Lu, E. J. Jang, Y. P. Lin, C. H. Hung, C. C. Chuang, and R. B. Lin "Simultaneous transistor pairing and placement for CMOS standard cells," DATE, pp. 1647-1652, Mar. 2015.