Area Minimization of Power Distribution Network Using Efficient Nonlinear Programming Techniques*

Xiaohai Wu[1], Xianlong Hong[1], Yici Cai[1], C.K.Cheng[2], Jun Gu[3] and Wayne Dai[4]

[1]Dept. Of Computer Science and Technology, Tsinghua University, Beijing, China, 100084

[2]Dept. Of Computer Science and Engineering, Univ. Of California at San Diego, La Jolla, California

[3]Dept. Of Computer Science, Hong Kong Univ. Of Science and Technology, Hong Kong

[4]Dept. Of Computer Engineering, Univ. Of California at Santa Cruz, California

E-mail:{wuxh, hongxl, caiyc}@tiger.cs.tsinghua.edu.cn

ABSTRACT: This paper deals with area minimization of power distribution network for VLSIs. A new algorithm based on efficient nonlinear programming techniques is presented to solve this problem. The experiment results prove that this algorithm has achieved the objects that minimize the area of power/ground networks with higher speed.

1. INTRODUCTION

Power/Ground (p/g) nets are always very important in design of VLSIs because they affect the performance of circuits seriously. Since their wire widths are much wider than signal nets, p/g nets usually cover a large portion of chip area. So they are often given the first priority in routing process.

There are two basic constraints in design and optimization of p/g nets. The first is the undesirable wear-out of metal wiring caused by electromigration, and the second is the narrowing margins caused by voltage drops[7][8]. Although increasing wire width can solve these problems, it would cost too much wiring resources. Consequently, it is necessary to minimize the area of p/g nets under the two constraints[8].

Generally speaking, the design of p/g nets consists of two main steps. Firstly, a topology for p/g nets is constructed, and it can be trees[4][10] or general graph[1][2][8]. Secondly, wire widths of p/g nets are minimized. It is not difficult to minimize wire widths of trees' branches because their current directions are fixed. However, to minimize wire widths of a general graph is much more difficult because current directions are uncertain before branch widths are determined. In most cases, it's a problem of non-linear optimization subject to non-linear constraints. Moreover, the number of variables is usually tremendous because of large scale of modern VLSIs.

Several studies on area optimization for p/g nets have been published. The algorithm presented by [8] is based on feasible direction method[12]. However, feasible direction method may bring about the problem of zigzagging[5], which leads to poor convergence.

Augmented Lagrangian function is used by [1]. In order to avoid computation of partial differential of currents subject to resistance, both currents and resistance are used as variables and Kirchoff's law is regarded as a constraint, which results in redundant searching space and increasing the scale of problem.

In 1999, Xiangdong Tan and C.-J. Richard Shi proposed an interesting algorithm[9]. The basic idea is to result a constrained nonlinear programming problem into a sequence of linear programming. Like [1], voltages and currents are also used as variables.

To solve these problems, we present a new algorithm based on penalty method, conjugate gradient method and circuits sensitivity analysis. Only conductance is used as variables. As a result, this algorithm is able to deal with real designs of IC industry.

2. PROBLEM FORMULATION

In the latter discussion, we assume that every component absorbs its most current, as is so called worst case. [2] has proved that if the constraints were not violated in worst cases they would be satisfied in other cases. For the similarity of power and ground nets, we will only describe the algorithm for power nets.During the description of problem formulation and solution method, following notations will be used:

E_{node} :set of indices of all nodes in p/g net.

E_{bch} :set of indices of all branches in p/g net.

E_{leaf} :set of indices of all power pins in E_{node}.

E_{vdrop} :set of indices of the nodes which violate voltage drop constraints in E_{node}.

E_{vemi} :set of indices of the branches which violate electromigration constraints in E_{bch}.

$E_{net}(p)$:set of indices of node p's neighboring nodes in E_{node}.

* This work is supported by National Science Foundation (No. 60076006) and 973 National Key Project (No. 1998030403)

ρ :sheet resistance.

l_{pq}, w_{pq}, g_{pq}:length, width and conductance of branch (p, q).

2.1 The Objective Function

The objective function is the area of p/g nets.

$$a = \sum_{(p,q)\in E_{bch}} l_{pq} w_{pq} = \sum_{(p,q)\in E_{bch}} \rho l_{pq}^2 g_{pq} \qquad (1)$$

Set $\alpha_{pq} = \rho l_{pq}^2$, $\quad a = \sum_{(p,q)\in E_{bch}} \alpha_{pq} g_{pq} \qquad (2)$

2.2 The Constraints

1) Voltage drop constraints

The difference between input voltage v_{dd} and power pins' voltages must be smaller than allowable biggest voltage drop u,

$$v_{dd} - v_i \geq u \qquad for\ all\ \ i \in E_{leaf} \qquad (3)$$

2) Current density constraints

$$\left|i_{pq}\right| / w_{pq} \leq \sigma \Rightarrow \left| v_p - v_q \right| \leq \rho l_{pq} \sigma \quad for\ all \quad (p,q) \in E_{bch} \ (4)$$

3) Kirchoff's law

Kirchoff's law can be described by the node voltage equations set: $\sum_{q\in E_{nei}(p)} (v_q - v_p) g_{pq} = i_p \quad for\ all \quad p,q \in E_{node} \qquad (5)$

In this algorithm Kirchoff's law is satisfied naturally and doesn't emerge explicitly.

4) Minimal width constraints

Owing to technological restriction, power branches have to be wider than minimal metal line width, which can be regarded as a constraint. However, a more simple way is to set the lines that do not satisfy this constraint as minimal metal line width, which is more efficient and leads to better results.

3. SOLUTION METHOD

3.1 Formulation of Penalty Function

Penalty method is the mainframe of this algorithm and penalty function is defined as below,

$$f = a + w \cdot \left(\sum_{i\in E_{leaf}} s_i^2 + \sum_{(p,q)\in E_{bch}} t_{pq}^2 \right) \qquad (6)$$

$$s_i = \begin{cases} 0 & if\ \ v_i \geq v_{dd} - u \\ v_i - (v_{dd} - u) & else \end{cases} \quad for\ all\ i \in E_{leaf} \qquad (7)$$

$$t_{pq} = \begin{cases} 0 & if\ \ \rho l_{pq}\sigma > \left| v_p - v_q \right| \\ \rho l_{pq}\sigma - \left| v_p - v_q \right| & else \end{cases} \quad for\ all\ (p,q) \in E_{bch} \qquad (8)$$

Where a is defined by (2) and w is penalty parameter.

Set $p_t = w \cdot \left(\sum_{i\in E_{leaf}} s_i^2 + \sum_{(p,q)\in E_{bch}} t_{pq}^2 \right)$ (9), penalty function can be rewritten as $f = a + p_t \qquad (10)$

3.2 Penalty method

At first, penalty parameter w is given an initial value, then w is increased and new penalty function is optimized again until all constraints are satisfied. In this way, original problem is transferred into a sequence of unconstrained minimization problems. The solution method can be described as below,

1. Set penalty parameter w as an initial value, initial conductance vector as $G^{(0)}$ and error bound as $\varepsilon_1 > 0$.
2. Solve unconstrained minimization problem, obtain current conductance vector $G^{(l)}$.

$$\min f = a + p_t = a + w \cdot \left(\sum_{i\in E_{leaf}} s_i^2 + \sum_{(p,q)\in E_{bch}} t_{pq}^2 \right) \qquad (11)$$

3. If $p_t < \varepsilon_1$, then stop, else increase w, and go to step 2.

In algorithm based on penalty function, the key is to find an efficient method to solve the unconstrained minimization problem. In our algorithm, *FR* conjugate gradient method is applied. Its steps are shown as follows:

1. Suppose initial conductance vector as $G^{(l)}$ and error bound as $\varepsilon_2 > 0$, initial descent direction is set as negative direction of gradient, $P^{(l)} = -\nabla f(G^{(l)}) \qquad (12)$
2. Determine a nonnegative scalar λ_k which minimize f. Let $G^{(k)}$ be the conductance vector at the kth iteration,
$$f(G^{(k)} + \lambda_k P^{(k)}) = \min f(G^{(k)} + \lambda P^{(k)}) \qquad (13)$$
3. Update conductance vector $G^{(k+1)} = G^{(k)} + \lambda_k P^{(k)} \qquad (14)$
4. If $\left\|\nabla f(G^{(k+1)})\right\| < \varepsilon_2$, then stop; else chose a new descent direction $P^{(k+1)}$ which satisfies the condition below,
$$\begin{cases} \beta_k = \dfrac{\left\|\nabla f(G^{(k+1)})\right\|^2}{\left\|\nabla f(G^{(k)})\right\|^2} \\ P^{(k+1)} = -\nabla f(G^{(k+1)}) + \beta_k P^{(k)} \end{cases} \qquad (15)$$
5. Go to step 2.

3.3 Line Search

After getting descent direction, we need to determine nonnegative scalar λ_k which minimize object function, that is,

$$f(G^{(k)} + \lambda_k P^{(k)}) = \min f(G^{(k)} + \lambda P^{(k)})$$

This process is what so called line search, which is used to reduce objective function along descent direction as much as possible. Since derivative of penalty function subject to λ_k is difficult to get, quadratic interpolation method is applied for it only requires the value of penalty function.

3.4 Resize Objective Function and Update penalty parameter

In optimization of p/g nets, the numerical difference between objective function and penalty term is very large, which would result in over consideration of objective function and make the process of optimization difficult to continue. Increasing penalty parameter will lead to ill-conditioning problem, a better solution is to resize objective function by a scalar γ. Define *wit* as the ratio of penalty terms to objective function and make it fixed, γ can be expressed by *wit* as,

$$wit = \frac{p_t}{\gamma \sum_{(p,q)\in E_{bch}} \alpha_{pq} g_{pq}} \Rightarrow \gamma = \frac{p_t}{wit \sum_{(p,q)\in E_{bch}} \alpha_{pq} g_{pq}} \qquad (16)$$

In traditional penalty method, w is always updated by timing a fixed updating ratio, which may let experiment results rely on updating ratio. To solve this problem, we use almost the same way which determines γ to update penalty parameter w. Set penalty parameter w in last unconstrained optimization is w_{old}, the new one w_{new} can be calculated by the constant wit as below,

$$wit = \frac{\frac{p_t}{w_{old}} \cdot w_{new}}{\gamma \cdot a} \Rightarrow w_{new} = w_{old} \cdot \frac{wit \cdot \gamma \cdot a}{p_t} \qquad (17)$$

4. GRADIENT CALCULATION

It is necessary to calculate the gradient of penalty function to get descent direction in conjugate gradient method. The following notations will be used in the description.

N: original network.

$N'(i)$, $N'(p)$, $N'(q)$: N's adjoint networks used to calculate gradient of voltages of nodes i, p and q.

n_{node}: nodes number of N or its adjoint networks.

n_{bch}: branches number of N or its adjoint networks.

M: coefficient matrix of N or its adjoint networks.

$V'(i)$, $V'(p)$, $V'(q)$: vectors formed by node voltages of $N'(i)$, $N'(p)$ and $N'(q)$ respectively.

$B(i)$, $B(p)$, $B(q)$: right side of node voltage equation sets derived from $N'(i)$, $N'(p)$, $N'(q)$.

v_i, v_p, v_q: voltages of node i, p and q in N.

G: conductance vector of N or its adjoint networks.

g_{ab}: conductance of branch (a, b) in N or its adjoint networks, $(a, b)\in E_{bch}$

v_a, v_b: the voltages of branch (a, b)'s two end nodes in N.

$v_a'(i)$, $v_b'(i)$: the voltages corresponding to v_a and v_b in $N'(i)$.

$v_a'(p)$, $v_b'(p)$: the voltages corresponding to v_a and v_b in $N'(p)$.

$v_a'(q)$, $v_b'(q)$: the voltages corresponding to v_a and v_b in $N'(q)$.

Since v_p and v_q have been known before each iteration, absolute value is unnecessary. Suppose $v_p > v_q$, (7) and (8) can be rewritten as,

$$\begin{cases} s_i = v_i - (v_{dd} - u) & for\ all\ \ i \in E_{vdrop} \\ t_{pq} = \rho l_{pq} \sigma - (v_p - v_q) & for\ all\ \ (p,q) \in E_{vemi} \end{cases} \qquad (18)$$

From (10), the partial differential of penalty function f subject to conductance g_{ab} can be expressed as,

$$\frac{\partial f}{\partial g_{ab}} = \frac{\partial a}{\partial g_{ab}} + \frac{\partial p_t}{\partial g_{ab}} \qquad (19)$$

With (2), (9) and (18), (19) can be rewritten as,

$$\frac{\partial f}{\partial g_{ab}} = \alpha_{ab} + w \cdot \left(\begin{array}{l} \sum_{i\in E_{vdrop}} \frac{\partial v_i}{\partial g_{ab}} \cdot 2 \cdot s_i + \\ \sum_{(p,q)\in E_{vemi}} \left(\frac{\partial v_q}{\partial g_{ab}} - \frac{\partial v_p}{\partial g_{ab}} \right) \cdot 2 \cdot t_{pq} \end{array} \right) \qquad (20)$$

(20) is difficult to calculate for node voltages cannot be expressed by conductance in explicit form. To solve this problem, we use the method of adjoint network, which is presented by *Director* and *Rhorer* in 1969[3][8]. Next we will describe how to calculate gradient of node i's voltage subject to conductance vector.

The first step is to build adjoint network $N'(i)$, which has the same topology and conductance vector with N. However, all absorbing current of $N'(i)$'s leaf nodes are set as 0 except node i, which absorbing current is set as $1A$. Besides, all input voltages are set as 0. The second step is to form node voltage equations set for $N'(i)$. Since N and $N'(i)$ have the same topology and conductance vector, they share the same coefficient matrixes. Their difference only resides in the right side: all elements of right side of $N'(i)$'s node voltage equations set are zero except the element corresponding to node i is –1, that is,

$$B(i) = [0,0,\cdots,0,-1,0,\cdots,0]^T$$

Where negative one appears at index i. So $\frac{\partial v_i}{\partial g_{ab}}$ can be calculated as, $\frac{\partial v_i}{\partial g_{ab}} = (v_a - v_b) \times (v_a'(i) - v_b'(i))$ (21)

With (21), (20) can be rewritten as,

$$\frac{\partial f}{\partial g_{ab}} = \alpha_{ab} + 2 \cdot w \cdot (v_a - v_b) \left(\begin{array}{l} \left(\sum_{i\in E_{vdrop}} v_a'(i) s_i + \sum_{(p,q)\in E_{vemi}} (v_a'(q) - v_a'(p)) t_{pq} \right) \\ -\left(\sum_{i\in E_{vdrop}} v_b'(i) s_i + \sum_{(p,q)\in E_{vemi}} (v_b'(q) - v_b'(p)) t_{pq} \right) \end{array} \right) \qquad (22)$$

Thus, the gradient of penalty function f subject to conductance vector G can be expressed as follow,

$$\nabla f(G) = \left[\frac{\partial f}{\partial g_1}, \frac{\partial f}{\partial g_2}, \cdots, \frac{\partial f}{\partial g_{ab}}, \cdots, \frac{\partial f}{\partial g_{n_{bch}}} \right]^T$$

(22) shows that we may have to solve an original network and many adjoint networks in each iteration to get $\nabla f(G)$, which will cost too much CPU time. Fortunately, we find a way to avoid solving each adjoint networks respectively,

$V'(i)$ can be got by solving node voltage equations set $MV'(i) = B(i)$, $v_a'(i)$ and $v_b'(i)$ are two elements of $V'(i)$, let vector $C_a = [0,0,\cdots,0,1,0,\cdots,0]$ and $C_b = [0,0,\cdots,0,1,0,\cdots,0]$ where 1 appears at index a and b respectively, we have,

$$v_a'(i) = C_a V'(i),\ v_b'(i) = C_b V'(i)$$

Then we can get $\sum_{i\in E_{vdrop}} s_i \cdot V'(i)$ and $\sum_{(p,q)\in E_{vemi}} t_{pq}(V'(q) - V'(p))$ by solving the following linear equation respectively,

$$M \sum_{i\in E_{vdrop}} s_i V'(i) = \sum_{i\in E_{vdrop}} s_i B(i) \qquad (23)$$

$$M \sum_{(p,q)\in E_{vemi}} t_{pq}(V'(q) - V'(p)) = \sum_{(p,q)\in E_{vemi}} t_{pq}(B(q) - B(p)) \qquad (24)$$

Add (23) and (24), $\sum_{i\in E_{vdrop}} s_i V'(i) + \sum_{(p,q)\in E_{vemi}} t_{pq}(V'(q) - V'(p))$ can be calculated by solving,

$$M \cdot \left[\sum_{i \in E_{vdrop}} s_i V'(i) + \sum_{(p,q) \in E_{vemi}} t_{pq}(V'(q) - V'(p)) \right] = \sum_{i \in E_{vdrop}} s_i B(i) + \sum_{(p,q) \in E_{vemi}} t_{pq}(B(q) - B(p)) \quad (25)$$

Use X to replace the vector on the left side of (25) and B_{new} to replace the vector on the right side of (25), (25) is rewritten as,

$$MX = B_{new} \quad (26)$$

Then, we express (22) as

$$\frac{\partial f}{\partial g_{ab}} = \alpha_{ab} + 2 \cdot w \cdot (v_a - v_b)(C_a - C_b)X \quad (27)$$

To obtain the gradient of penalty function subject to G, we should merge adjoint networks by forming a new right side according to (25), then solve (26), and finally use (27) to get $\nabla f(G)$. Thus the gradient of penalty function subject to conductance vector can be obtained at one time. Since the coefficient matrix derived from node voltage equations set is symmetric and positive definite, Incomplete Cholesky Decomposition Conjugate Gradient (ICCG)[6][11] is used to solve node voltage equations set.

In ICCG, computing pre-conditioned matrixes will costs most time. As the coefficient matrixes of original and adjoint network are the same, they can share the same pre-conditioned matrix. Besides, most p/g nets have very special topology; we can use the equivalent circuit technique to accelerate p/g net solver's speed.

5. Analysis of Time Complexity

To discuss time complexity, we have such notations:

N_{iter} :Number of iteration.

$N_{line}(i)$:Number of line searching in iteration i.

$T_{line}(i, j)$:Time of solving a network during line searching j in iteration i.

$T_{pre}(i)$:Time of computing pre-conditioned matrix for ICCG in iteration i.

$T_{so}(i)$:Time of solving original network with pre-conditioned matrix in iteration i.

$T_{sa}(i)$:Time of solving (26) with pre-conditioned matrix in iteration i.

In each iteration we need solve original network and (26) to get new searching direction, then do line searching. So time complexity can be expressed as below,

$$T = \sum_{i=1}^{N_{iter}} \left[T_{pre}(i) + T_{so}(i) + T_{sa}(i) + \sum_{j=1}^{N_{line}(i)} T_{line}(i, j) \right] \quad (28)$$

According to (28), the algorithm's efficiency depends on the speed of p/g net solver. Generally speaking, the complexity of solving a linear equations by iteration method is $O(\#iter \times N^2)$, where N is the range of coefficient matrix. However, the coefficient matrix of node voltage equations set is symmetric, positive definite and very sparse. So ICCG can solve it much faster. Besides, most p/g nets have a very special topology described in section 4, which can accelerate p/g net solver's speed. So it is difficult to get the exact complexity of solving a p/g network. Instead, we present several examples to illustrate empirical trend of our p/g nets solver's efficiency as table 1 shows,

Table 1. Examples to illustrate p/g net solver's efficiency

circuit name	*node number*	*time*(*s*)
test1	1309	0.01
test2	3741	0.07
test3	7492	0.14
test4	32112	0.30
test5	48168	0.53
test6	112392	0.84
test7	160560	1.08
test8	321120	2.08
test9	1618026	7.85

Table 2. Comparison of our algorithm against Tan-Shi algorithm

circuit name	*node number*	*branch number*	*our algorithm*		*Tan-Shi algorithm*		*speed up*
			time (*s*)	*area reduced* (*%*)	*time* (*s*)	*area reduced* (*%*)	
p4x4	17	23	2.34	99.853	0.43	95.1	0.18
p3x500	1502	1505	3.67	49.5234	37.6	47.8	10.2
g300x10	3002	3599	57.34	99.353	609.9	93.7	10.6
p100x100	10002	10199	87.43	98.749	1325.6	80.7	15.2

Table 3. The experiment results of larger circuits

circuits name	*node number*	*straps number*	*trunks number*	*area*(μm^2)		*time*(*s*)
				branch-and-bound method	*our algorithm*	
test1	1309	30	3	8971.80	6733.41	3.55
test2	3741	61	5	21540.48	17477.02	15.07
test3	7492	87	9	72380.88	70035.05	58.43
test4	32112	186	8	76789.44	64417.61	88.78
test5	48168	228	5	38599.68	24279.34	101.41
test6	112392	348	6	58936.22	51869.75	184.92

test7	160560	417	2	35228.16	24051.18	257.71
test8	321120	688	6	152662.73	84971.14	2206.62
test9	1618026	1577	5	383351.25	358620.66	6392.67

6. EXPERIMENT RESULTS

This algorithm has been developed on *sun ultra_sparc* 250M workstation with language C and C++. For comparison, we have used the test cases which [9] presented. The results of comparison are shown as table 2.

In table 2, the column of *area reduced*(*%*) is reduced wiring area of original area in percentage, which shows that all p/g nets optimized by our algorithm occupy smaller area than [9]. The column of *speed up* says that to large circuits like p100x100 our algorithm is much faster than Tan-Shi algorithm.

There are two reasons for area improvement. Firstly, only conductance is looked as the variables, which avoids redundant searching space. Secondly, there is no any assumption such as fixing currents or voltages like *Tan-Shi* algorithm.

We have also tested this algorithm with much larger circuits from IC industry. To compare with this algorithm, we have also developed an algorithm based on branch-and-bound method to optimize p/g nets area. Table 3 lists the results.

Comparing with other algorithm, our algorithm is able to deal with large circuits. There are 1,618,026 components in our largest test circuit. The running time is 6392.67 second. From table 3, we can see that area obtained by the algorithm proposed in this paper is much smaller than that derived from branch-and-bound method.

Figure 1 shows how the number of the nodes, which violate the voltage drop and the electromigration constraints, decreases with iteration (take circuit test1 as example). The convergence characteristic of this algorithm is also described by figure 1. The vertical lines say that penalty parameter is updated and a new unconstrained optimization begins. At these times the number of violating nodes decreases more sharply and at last it converges to zero.

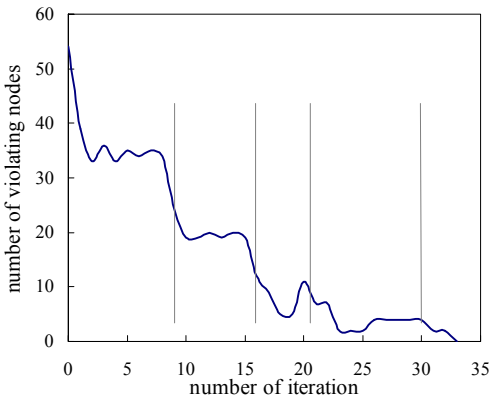

Fig. 1. Violating nodes reduction with the number of iterations

7. CONCLUSION AND FUTURE WORK

In this paper we present a new algorithm based on penalty method, conjugate gradient method, circuits sensitivity analysis and merge of adjoint networks to minimize the area of p/g nets while satisfying the constraints. The experiment results show that this program is very robust with high speed and low wiring resource consumption. Further more, it has the capability of optimizing large-scale circuits.

REFERENCE

[1] S. Chowdhury and M. A. Breuer, "Minimal area design of power/ground nets having graph topologies," IEEE Trans. on Circuits and Systems, pp.1441~1451, December. 1987.

[2] S. Chowdhry and M. A. Breuer, "Optimization Design of Reliable IC Power Networks Having General Graph Topologies," Proc. 26th DA Conf., pp. 787~790, 1989.

[3] S. W. Director and R. A. Roher, "A Generalized Adjoint Network and Network Sensitivities," IEEE Trans. on Circuits and Systems, Vol. CT-16, pp. 318~323, August. 1969.

[4] K-H. Erhard, F. M. Johannes and R. Dachauer. "Topology Optimization Techniques for Power/Ground Networks in VLSI," Proc. European Design Automation Conference, pp. 362~367, 1992.

[5] R. Fletcher, "Practical Method of Optimization," Vol.2, New York: Wiley, 1981.

[6] Gene H. Golub and Charles F. Van Loan, "Matrix Computations," Johns Hopkins University Press, 1983

[7] F. M. D'Heurle, "Electromigration and failure in electronics: An introduction," Proc. IEEE, vol. 59, pp. 1409~1418, Oct. 1971.

[8] Mitsuhashi T. & Kuh E. S, "Power and Ground Network Topology Optimization for Cell Based VLSIs", proceedings of 29th ACM/IEEE Design Automation Conference, pp. 524~529, 1992.

[9] Xiang-Dong Tan, C.-J. Richard Shi, Dragos Lungeanu, Jyh-Chwen Lee and Li-Pen Yuan, "Reliability-Constrained Area Optimization of VLSI Power/Ground Networks Via Sequence of Linear Programmings", DAC 99 New Orleans, Louisiana, pp. 78-83, 1999.

[10] WU Xiao-hai, QIAO Chang-ge, YIN Li, HONG Xian-long，"Design and Optimization of Power/Ground Network for BBL-Based VLSIs"， August 2000, Acta Electronica Sinica.

[11] Xiaohai Wu, Li Yin, Xianlong Hong, "A Power and Ground Network Solver with the Method of Incomplete Cholesky decomposition Conjugate Gradien", March 2000, Chinese Journal of Semiconductors.

[12] G. Zoutendijk, "Methods of Feasible Directions," Amsterdam, The Netherlands: Elsevier, 1960.