第5卷 第4期 半 导 体 学 报 Vol. 5, No. 4
1984 年 7 月 CHINESE JOURNAL OF SEMICONDUCTORS July, 1984

快速最优通道布线算法

吴 祖 增
(复旦大学电子工程系)

1983 年 7 月 20 日

本文提出的快速最优通道布线算法，是 Kernighan-Shweikert-Persky 最优通道布线算法和 Wada 曲干最优通道布线算法的改进. 这些算法的"最优性"意义完全相同，但新算法的执行速度统计地比上述二法远远为快.

一、引 言

通道布线算法是大规模集成电路和印刷电路板布线设计中广泛应用的一类算法，其特点是采用并行布线方式，因而与串行布线的李氏算法和 Highitower 算法相比，可以获得较高的整体布线质量并容易处理 100% 布线率问题.

通道布线算法最早由 Hashimoto-Stevens[1] 提出. 他们的算法应用于无约束线网集可保证获得最优介，但应用于约束线网集时结果常不理想. 后来，Kernighan 等人[2]在此基础上应用分支限界法解决了无环约束集的最优布线，但算法过程较长. Wada[4] 则把分支限界法推广到曲干布线[3]，并提出了一个解决带环约束集 100% 布线的方案，此外，还着重介绍了两个分支限界过程的加速措施. Wada 的两个措施是有益的，但试验表明时间仍可观(见[4]结束语). 本文就是在 Wada 工作基础上进一步采用了若干加速措施. 大量试验表明这些措施是有效的，多数例子在联合应用各种措施后分支限界过程获得了根本简化，且其余的也有不少简化.

二、基 本 思 想

分支限界法的执行过程可分为两个阶段：①在判定树上依次搜索，直到获得一个最优解；②验证此解的最优性，即验证其后一切分支上不再有更优解. 显然，要缩短整个算法过程就必须同时缩短搜索和验证两个过程. 以下是快速最优通道布线算法采用的几个措施：

1. 线网重排次序 在分支限界过程开始之前，将线网重新排队，使之有利于尽快搜索到最优解和较优解；

2. 分层次优化 先用一个快速优化算法来产生一个高质量的初始布线，然后再用一般分支限界法进一步优化；

3. 快速最优性验证 即除了利用分支限界法外，还使用一种简单但常常十分有效

的手段来验证已获得解的最优性；

4. 用增量法计算界限　即除了首次外，分支限界过程中任何一次求界都是在上一次求界所得数据基础上进行的；

5. 寻找更精确的界限　寻找一个比 Kernighan 的动态界限能更精确估算布线需要行数的界限来替代动态界限.

合理的线网排列，优良的初始布线，快速和精确的求界手段都有利于缩短上面所说的第一阶段，而有效的验证手段以及快速精确的求界方法则有助于缩短第二个阶段．实际上，只要不是人为的例子，应用快速最优通道布线算法，在大多数情况下，都能快速获得最优解，并快速验证其最优性，用不到像一般分支限界过程那样进行耗尽式的搜索和验证.

三、加速技巧

以下介绍上述措施的细节.

1. 线网重排次序

这一措施的目的是让最优解或较优解尽可能地排列在判定树较早搜索到的那些分支上．判定树的结构由分支法则确定，Kernighan 算法的分支法则就是左边法则(在每一行中，从左到右，每次总是选择当时的最左可布线网布之)．但因采用分重叠区（Zone）结构后最左可布线网一般不是唯一的，因而选择其中哪一个最左可布线网先布就有灵活性，这就使判定树的结构在一定程度上可以改变．本措施的具体做法是：在线网的静态界限和约束级别求出之后，在每个重叠区中，将所有左端位于该区的线网按照静态界限大小由大到小排列，如果静态界限一致，则按约束级别高低排．进入分支限界过程后，每次都按这个顺序作为选择最左可布线网的优先顺序.

这一措施可使部分例子只用 Hashimoto-Stevens 的左边算法就能获得最优解.

2. 初始布线

可以用任何快速优质通道布线算法来产生初始布线，以下是两个这样的算法.

A. 双边算法 (BEA)

见图 1．它和左边算法(下记为 LEA)一样，采用从上到下(或从下到上)一行行地布线．但在每一行中，一开始总是选择当时具有最大静态界限和最高约束级别的未布线网布于该行的"中间"，然后，在此线网右边应用左边算法布线，直到通道右端；而在此线网左边，则应用右边算法布线，直到通道左端．同一重叠区中若有几个线网可选择，则在布线之前所作的线网重排(框 3、4)能保证优先选择静态界限大的和约束级别高的先布.

双边算法使半数以上的试验例子获得最优解，且其余的也非常接近最优解.

B. 倒限界分支限界法 (RBT)

这是一种限界次序与一般完全颠倒的分支限界法．一般分支限界法为了保证获得解一开始必须假设一个很宽的限界值，然后让在分支限界过程中实际得到的当前最优解值来不断更新，最终达到最优值.

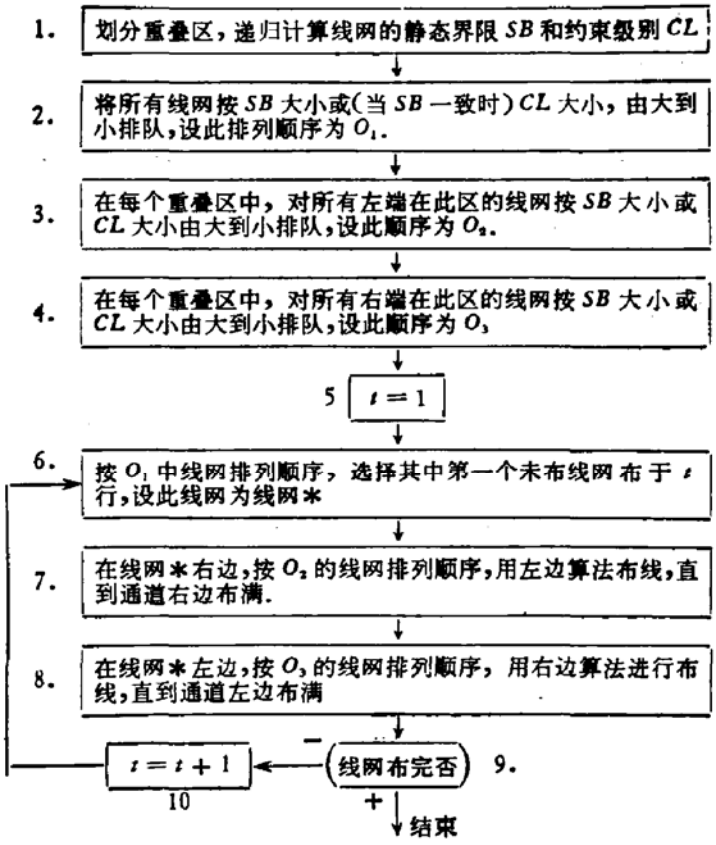


图 1 双边算法

这一过程有时会很长，这就是在判定树上开始都是一些差的解，逐步出现较优的解，而最优解排在很后面.

考虑到这种可能情况，倒限界法一开始就用一个下界作为界限初值，以达到快速剪支的目的，如果发现实际解都达不到这一数值，则再放宽它，这样由紧到宽来逼近最优解.

倒限界法可以用作整体优化，亦可用作局部优化. 作为初始布线算法，以下介绍一个逐行应用分支限界法而整个不用的快速部份优化倒限界法. 基本思想如下：

一开始，令 t (行数) $=1$，V (限界值)=线网全集 S 的动态界限 $DB(S)$. 然后，从上到下一行一行用 LEA 布线. 当一行布满时，计算未布集的动态界限 DB 并判别关系式 $OB+t\leqslant V$ 是否成立，若成立，则 $t+1\rightarrow t$，去布下一行；否则就按后布先拆的顺序，将 t 行上位于右边的部分线网拆除(拆除规则同 Wada[4]，详见框图)，并用某些最左可布线网继续布满 t 行 (选择规则同 Wada[4]，详见框图)，再计算未布集的 DB 并判别关系式 $DB+t\leqslant V$ 是否成立，……，这样不断用尚有可能满足关系式的各种可行布线来试布该行，最后有两种可能：①在某种布线下 $DB+t\leqslant V$ 关系式成立，这时就 $t+1\rightarrow t$，去布下一行；②一切可行布线都不能使 $DB+t\leqslant V$ 成立，这时就取本行各种可行布线中性能最好的一种，即能使未布集动态界限达到最小值 minDB 的那种布线作为本行布线，并令 $V=t+\min DB$，然后再 $t+1\rightarrow t$，去布下一行. 这样一直下去直到一切线网布完就结束.

图 2 为该算法框图. 其中 s 为已布线网数. Z 为当前布线起始 Zone. ZL，ZR 为

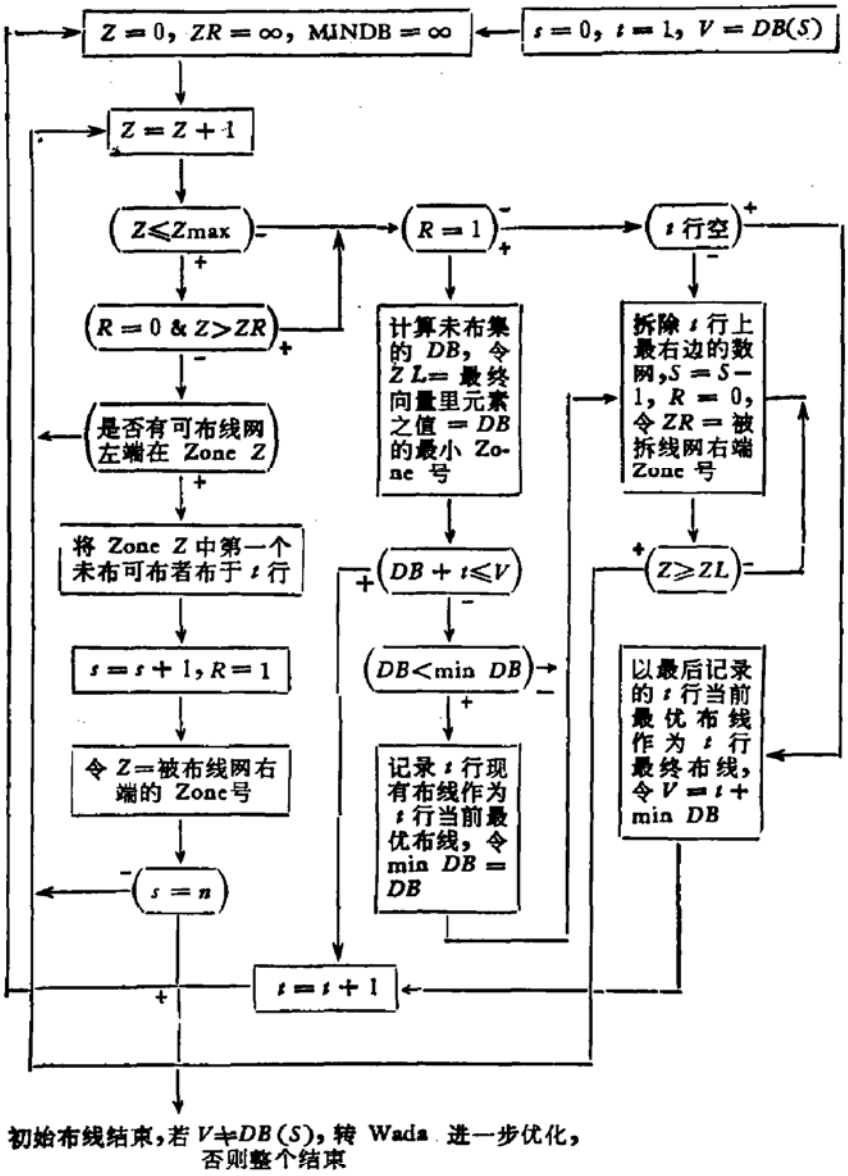


图 2　倒限界部分优化分支限界法

实现 Wada 加速技巧引进的参数．R 是一个标志，取 1 时代表当前为布线，取 0 为拆线，n 为线网总数．

图 3 是上述算法与 Kernighan、Wada 算法分支限界过程的比较．其中 (A) 为被布线网集 S，$DB(S)$ 等于 3．(B) 为线网约束图．(C) (D) (E) 为三个算法的分支限界过程．矢量向右为布线，向左为拆线．一个圆圈代表一次求界，圆内数字为已布行数与未布线网动态界限之和 $t + DB$．虚线与实线一起构成完整的判定树．(F) 为三算法的共同结果．

由图可知，同样达到最优解 ($V = 3$)，倒限界法要快得多．自然，作为部分优化算法它不保证都能获得最优解．因此一般地说需要进一步优化．但实际应用中大量例子都能获得最优解．从理论上则可以证明，只要每次求界都是精确值，则上述算法一定有最优解．

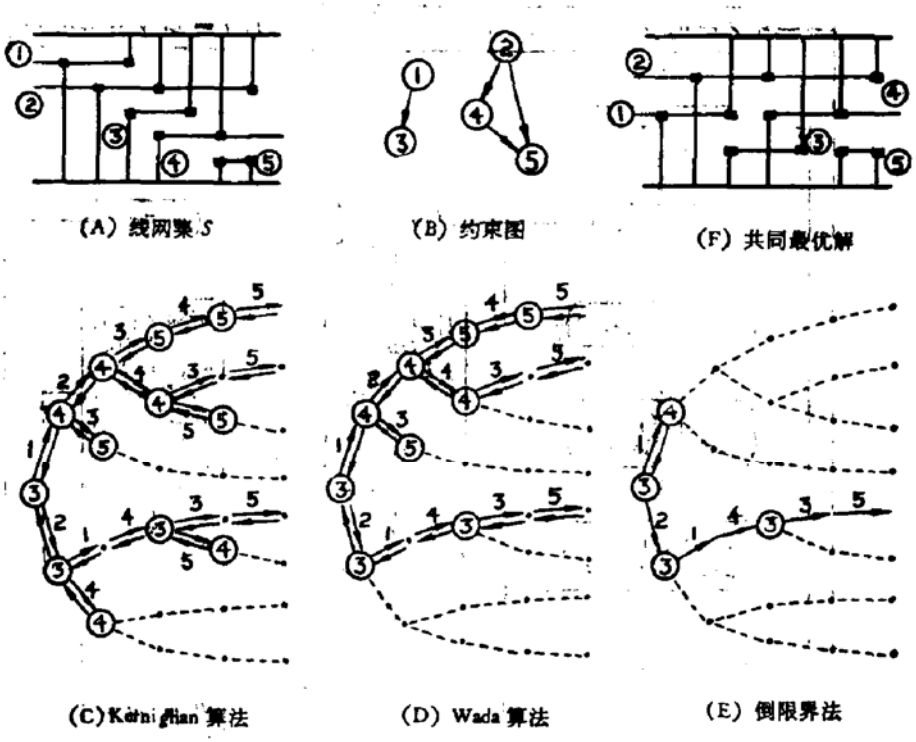


图 3 分支限界过程的比较

3. 快速最优性验证

当最优解达到之后，分支限界法本身并不能断定其为最优，必须在判定树上继续搜索，直到其后一切分支都考察完毕，发现没有更优解时，才能回头来肯定原来那个解已是最优解．这种方法对于可行解数且很大的所谓"大边缘"问题要消耗大量时间．例如我们曾用一个包含 50 个线网的无约束集作过试验，采用 Wada 方法．在布线 50 次，求界 9 次后即得到最优解，而验证其最优性却要在布线和拆线各 149624 次求界 130824 次后才完成．快速最优性验证方法很简单：每当得到一个当前最优解后，就检查它的值是否等于整个集合的动态界限，若相等，则因动态界限必小于等于最优解值，故可断定解已最优，不必再继续分支限界下去．自然，如果最优解值实际小于上述动态界限值，则此快速检验法无效，就必须进行完全的分支限界过程． 但这种情况实际是极少遇见的． 这一措施 Kernighan 后来已提到，但 Wada 没有采用．

4. 增量计算动态界限

动态界限的计算中要求构造一个二维数组(密度矩阵)，它的每一个元素又必须靠计数得到，因而工作量很大．为了压缩求界工作量，分 Zone 是明显有益的．另外，我们采用了增量计算法，收效也很可观．增量计算的思想是：除第一次，即整个集合的动态界限必须按定义直接计算外，其后任何一个部分集的动态界限都可利用最近一次动态界限计算中所得的数据经过修改得到．具体做法是：每次求界后，保留其密度矩阵和向量序列；当继续布线拆线时，同时也就对密度矩阵进行修改，并把更动线网的最小静态界限 K 记录下来；当再次求界时，只需利用修改后的密度矩阵对原向量序列中第 K 个及 K 以后的那部

分向量重算即可.

5. 更精确的求界方法

Kernighan 的动态界限用来估算布线需要的行数，极大多数情况下是完全精确的，但也有例外，图 4(A) 就是一个.该集合的动态界限为 7，而最优布线必须 8 行.(证略.)分析其原因，在于动态界限计算中没有考虑相互重叠的线网在布线过程中带来的新约束关系.例如在图 4 中，线网①与⑨或②与⑥原来都无约束关系. 但因①②水平重叠，在实际布线中，不是①在②上就是②在①上. 这样也就是说，或者①约束⑨，或者②约束⑥，二者必居其一.

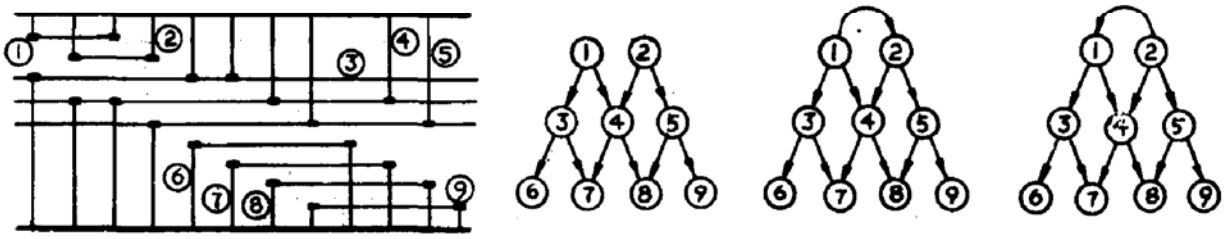


图 4　新的动态界限计算法

为了提高精度，新的动态界限要考虑这种情况. 对于上例，采用的办法是：从原线网集 S 形成两个新集 S_1 与 S_2，在 S_1 中令①约束②，而 S_2 中②约束①(图 4(C)，(D)) 然后分别计算 S_1 与 S_2 的动态界限 $DB(S_1)$，$DB(S_2)$. 而新的动态界限

$$NDB(S) = \min\{DB(S_1), DB(S_2)\}.$$

按此法计算，S 的界限等于8，与最优解值相等了.

必须说明，要完整处理线网重叠引起的垂直约束十分复杂，必须简化才行. 这就是只考虑静态界限大的且位于约束图顶部的少数线网，它们的相互重叠最有可能引起新约束. 另外，本措施宜选择使用，因对大多数实际问题来说旧界限已精确，无条件用新界限反而使问题复杂化.

四、快速最优通道布线算法

将上述措施中的一个或相容的几个加入 Kernighan 算法或 Wada 算法中就得到各种不同形式不同程度的快速最优通道布线算法 (FOCR)，例如以下形式都是可取的：

$FOCR_1$ = Wada 算法＋增量求界＋快速最优性验证；

$FOCR_2$ = $FOCR_1$ ＋ 线网预排序＋部分优化 RBT 预布线；

$FOCR_3$ = $FOCR_1$ ＋ 线网预排序＋完全优化的 RBT 法；

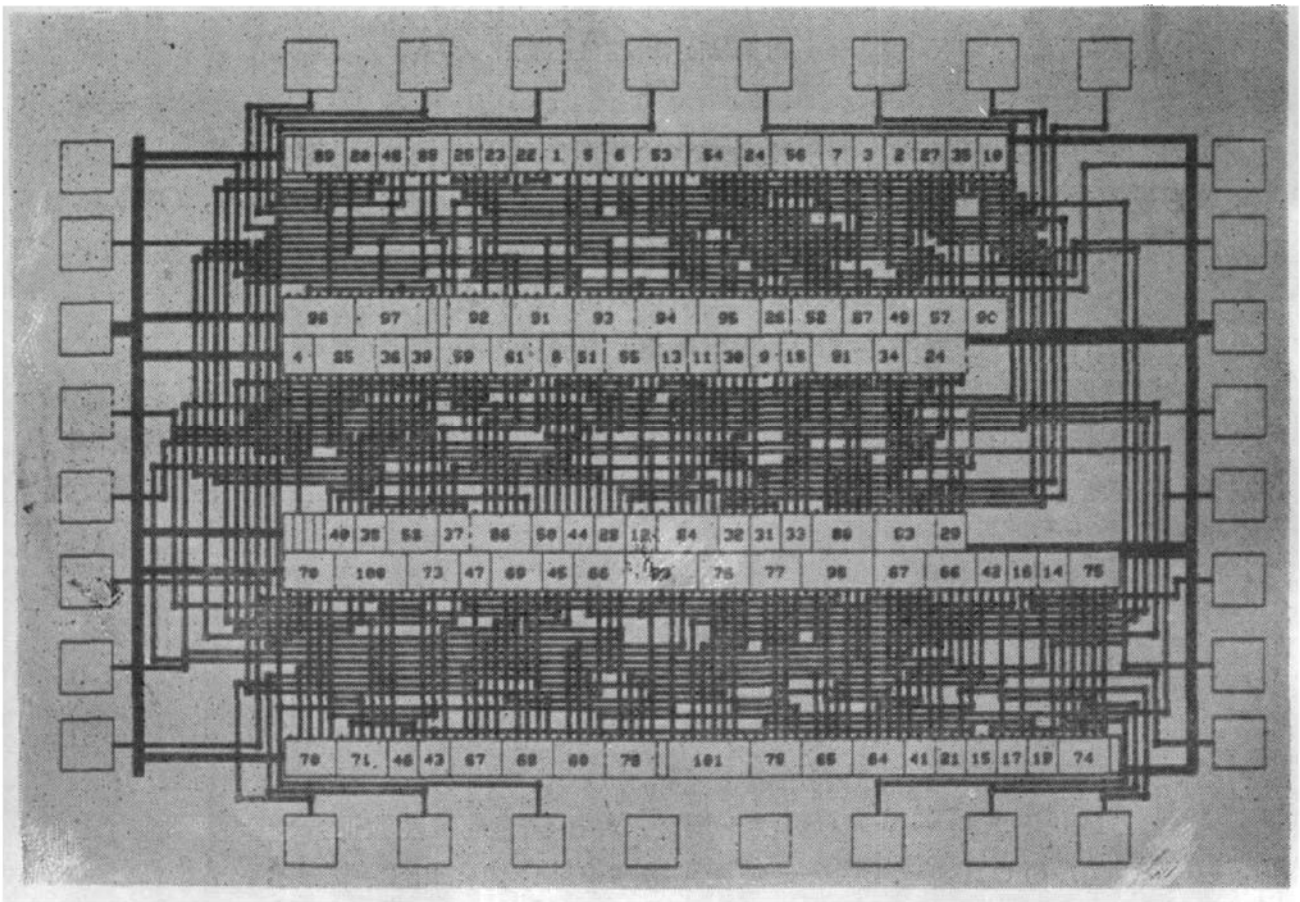
$FOCR_4$ = $FOCR_1$ ＋ 线网预排序 ＋ BEA 预布线.

上述算法我们编成 FORTRAN 程序后曾在 PDP11/34 上对 Kernighan 的几个例子[2]以及 Deutsch 的“困难问题”[4]进行了试验. 表 1 为其部分结果 ($FOCR_{3,4}$ 与 $FOCR_2$ 略同，

故省去). 这里用求界次数作为比较项目，这与 Wada 的“TRIAL”数一致. 所用布线顺序 Kernighan 例子 (1—6 列)全为从上到下从左到右; Deutsch 例子则因用 Wada 算法按上顺序运行 2 小时后无结果改用了从上到下从右到左顺序. 表 1 中即为该顺序下的结果，所用时间 Wada 法为 2069 秒，$FOCR_2$ 为 136 秒. 对此特例还以 Wada 本人采用的顺序，即从下到上从左到右作了试验，结果 Wada 法为 167 秒*，$FOCR_2$ 为 127 秒. 值得注意，这一问题用不同顺序布线时困难程度相差甚远. 为了减少运行时间，可选择一个“容易的”顺序布线(可用左、右边算法试布一下，看哪个结果最好就用那个顺序)，并利用不同顺序下的动态界限来选出最精确(大)的值. 加入这些措施后我们再试验了一次，结果 $FOCR_2$ 减到 36 秒，其中分支限界过程为 9 秒，其余都化在准备工作上.

表 1 最优算法求界次数比较

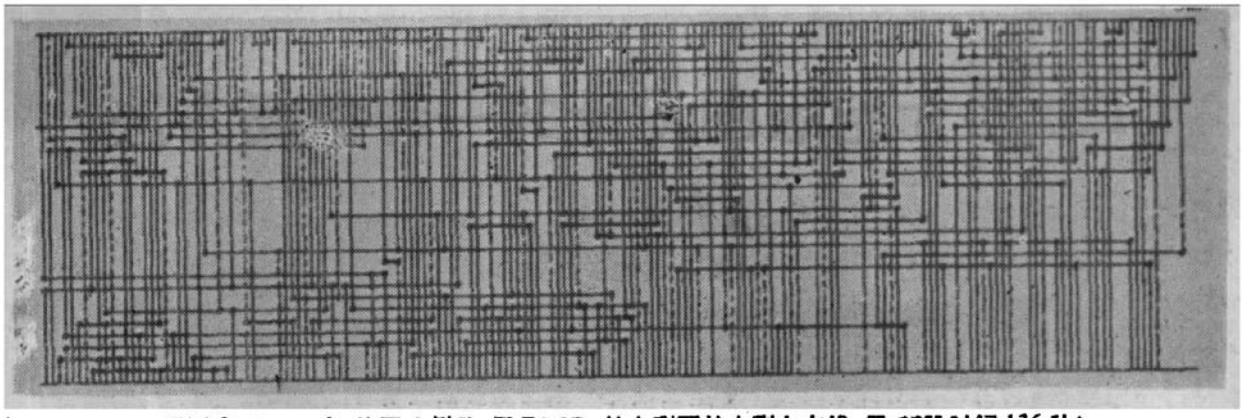
算 法	例 1	例 3 上	例 3 中	例 3 下	例 4b	例 5	Deutsch
Wada	112**	865	461	565	1241	7025	12998
$FOCR_1$	24	48	300	220	511	635	12896
$FOCR_2$	13	48	296	17	191	20	836



附图 1 Kernighan 的例 3，用 $FOCR_2$ 布线结果，三个主要通道共用 CPU 时间 16秒

* Wada 本人在 DEC-20 上试验用了 430 秒；这一差别和线网原始排列次序有关. 另外，我们的 Wada 算法中已采用增量法求界.

** Wada 本人按从下到上从左到右顺序试验时为 207 次.



附图 2　Deutsch 的困难例子，用 $FOCR_2$ 从上到下从右到左布线，用 CPU 时间 136 秒；采用自动选择最优布线顺序后则降为 36 秒

附图 1，2 为 $FOCR_2$ 对例 3 及 Deutsch 问题的布线结果。

五、结　论

通道布线问题属于 NP 困难问题，任何有效算法都不能保证结果的最优性，而任何最优算法都不能保证计算的高效性；快速最优算法无疑只是相对地快，统计地快．不排除遇到困难的可能性，但这种可能性已大大减少．事实上，在我们应用快速最优算法进行 LSI 和 PCB 的设计实践中还从未遇到过这样的问题，在总数接近 200 个的通道布线实例中，极大多数是在 1—2 秒内解决的，最长的也不过十多秒．

作者感谢唐璞山同志以及原复旦 CAD 组王世民、沙酪、陆昉等同志，本算法的某些思想以及程序细节是在这个组 80 年度讨论班上交流后完善起来的．还感谢计算机房同志的配合和协助．

参　考　文　献

[1] A. Hashimoto and J. Stevens, Proc. of 8th DAC., pp. 155—169 (1971).
[2] B. W. Kernighan, D. G. Schweikert and G. Persky, Proc. of 10th DAC., pp. 55—59 (1973).
[3] D. N. Deutsch, Proc. of 13th DAC., pp. 425—433 (1976).
[4] M. M. Wada, Proc. of 18th DAC., pp. 762—768 (1981).
[5] 沙酪、唐璞山，半导体学报，**5**，(1984)

Fast Optimal Channel Routers

Wu Zuzeng
(Fudan University)

Abstract

This paper presents some optimal channel routers based upon the optimum channel router and the dogleg "optimal" channel router, but improved to various degrees. All these routers have the "optimality" with the same sense, yet the speed of the new algorithms are statistically far much greater than that of the originals.