第 23 卷第 2 期 半 导 体 学 报 Vol. 23, No. 2
2002 年 2 月 CHINESE JOURNAL OF SEMICONDUCTORS Feb., 2002

A Fast and Efficient Global Router for Congestion Optimization*

Xu Jingyu[1], Bao Haiyun[1], Hong Xianlong[1], Cai Yici[1], Jing Tong[1] and Gu Jun[2]

(1 *Department of Computer Science & Technology, Tsinghua University, Beijing 100084, China*)
(2 *Department of Computer Science, Hong Kong University of Science & Technology, Hong Kong, China*)

Abstract: An efficient parallel global router using random optimization that is independent of net ordering is proposed. Parallel approaches are described and strategies guaranteeing the routing quality are discussed. The wire length model is implemented on multiprocessor, which enables the algorithm to approach feasibility of large-scale problems. Timing-driven model on multiprocessor and wire length model on distributed processors are also presented. The parallel algorithm greatly reduces the run-time of routing. The experimental results show good speedups with no degradation of the routing quality.

Key words: global routing; congestion optimizing; global routing graph (GRG); parallel algorithm
CCACC: 7410D
CLC number: TN47 **Document code**: A **Article ID**: 0253-4177(2002)02-0136-07

1 Introduction

Global routing plays an extremely important role of VLSI (very large scale integration) layout design[1,2]. Congestion optimizing in global routing is still a critical problem to be well solved[3]. In VLSI circuits, with the development of deep submicron and multi-layer metal wiring technology, chip density is undergoing a sharp increase. As a result, it is necessary for the congestion optimization algorithms to be more efficient to deal with problems of larger circuit scale with higher speed. Parallel processing technology is the efficient way to speed up one's algorithm, which has been widely used in the research field of integrated circuit (IC) computer aided design (CAD). Many research working on the parallel layout algorithm can be found in the proceedings of ICCAD (international conference on computer aided design) and DAC (design automation conference) from 1985 to 1990. Those works include the session of circuit simulation, circuit floorplanning and placement, system verification, test pattern generation and layout extraction. There are more than 30 useful papers for those proceedings. However, only a few papers have been found, and a limited progress has been reported on the parallel global routing algorithm. The main reason is that there is a strong seriality in the process of global routing. It is very difficult to design the parallel algorithm for global routing. We noticed that re-routing is time consuming and induces markedly increasing run-time for large-scale circuits in global routing. To eliminate

* Project partially supported by the National Natural Science Foundation of China (No. 60076016), 973 National Key Project of China (G-1998030403), and the Postdoctoral Science Foundation of China (No. [2000]23)

Xu Jingyu female, was born in 1976, PhD candidate. Her current research interest focuses on routing technology in IC physical design, especially performance-driven routing.

Bao Haiyun male, was born in 1972, PhD candidate. His current research interest focuses on routing technology in IC physical design, especially performance-driven routing.

Hong Xianlong male, was born in 1940, professor. His research interests include layout algorithms and systems.

Received 19 June 2001, revised manuscript received 28 September 2001

this bottleneck, we propose an efficient parallel global routing algorithm in this paper.

The remainder of this paper is organized as follows. Section 2 gives introduction of related research. Section 3 presents the parallel mechanism and the basic serial global routing algorithm. In section 4, we describe the proposed approach of our fast parallel router. Finally, section 5 gives experimental results and section 6 concludes the paper.

2 Related research

Several parallel global routers have been proposed. Using the graph search approach, a parallel algorithm was implemented on a hypercube-base distributed memory multiprocessor[4]. Parallel algorithms have also been described in References[5 ~ 7] that are based on an iterative improved approach. A hierarchical parallel algorithm was implemented on shared memory multiprocessor[8]. But those parallel algorithms have the following disadvantages: they were either targeted for a specific parallel architecture such as shared memory multiprocessors, or could only handle two-pin nets, or the qualities of the routing were poor compared with the state-of-the-art routers.

In the following sections, we will introduce our strategies in detail for parallel algorithm design and the basic serial algorithm.

3 Parallel mechanism and basic serial algorithm

3.1 Parallel mechanism

To guarantee the quality of the final parallel routing solutions, the following strategies are proposed: (1) In the parallel algorithm, we use the linear combination of total congestion of the passed edges, total wire length and the number of bends in one routing tree, denoted by E_t, to evaluate the routing tree of a net. Among all routing trees that have been constructed for a congested net, the one with minimal E_t is selected as the re-routed tree, which is not necessarily the current constructed one. As a result, each net can detour congested areas using suitable routing trees. In this way, the parallel algorithm can resolve routing resource conflicting in a more flexible way, control the total wire length to some extent and locally decrease the overflow. (2) In the parallel algorithm, a subset of congested nets N_{open} is randomly selected to reroute. The proportion of the net number in N_{open} to the total net number is decreased in a wavy mode. Therefore, estimation of the congested area can gradually approach the real value. (3) In the parallel algorithm, after iterations of the ConcurrentRouter, we will get a temporary result U_{good}. Then, U_{good} is improved by the SequentialRouter for a further local optimal solution. In parallel re-routing, inaccuracy in the weights of GRG edges is local, which will be adjusted later in updating. The whole process is essentially searching in the potential solution space, which guarantees the medium routing unsatisfactory results will not last long time in the whole routing process and the dead-loop can be avoided.

The implementation of our parallel global routing algorithm has the following advantages: (1) It is of high degree of parallelism. It can deal with large-scale circuits well. (2) Compared with rip-up and re-route algorithms, the quality of solution is independent of the routing order of the nets. (3) Compared with the multi-commodity flow model, it avoids integerization stage containing net ordering. (4) Compared with simulated-annealing method[9,10], such as the state-of-the-art global router TimberWolf[11], our algorithm reduces runtime by purposive control in randomization.

3.2 Basic serial algorithm

An efficient serial global routing algorithm is necessary for our parallel design. Firstly, a high degree of parallelism depends on the serial algorithm structure. Secondly, based on an efficient serial routing algorithm, the parallel routing algorithm

will be more powerful in routing ability. The above parallel strategies are based on our serial global routing algorithm RINO.

In the research field of serial global routing algorithms, many typical strategies have been proposed. They are sequential routing and re-routing[12, 13], iterative improvement method[14, 15], integer linear programming[16], hierarchical routing[17, 18], simulated evolution technology[19], feature routing[20, 21] and high-speed multilevel staged clock routing[22]. The integer linear programming and hierarchical routing method have comparatively high complexities for large-scale problems, which need improvement in calculability. The rip-up and reroute algorithms are widely adopted for their practicality and effectiveness[23], yet net ordering and non-predicting of congested areas are still two critical problems[3]. Our serial global routing algorithm RINO[24] solves these problems efficiently by using random optimization technology. Compared with the widely adopted global router Carden IV-Router[25] using multi-commodity flow method, RINO gets solutions with equivalent quality and runs much faster.

However, it takes a longer time for RINO algorithm to route the large-scale circuit, which will limit its routing ability. Based on RINO algorithm, we propose a fast parallel routing algorithm. Thus, we can speed up the routing algorithm and improve its routing ability on large-scale circuits. Meanwhile, we make full use of the advanced routing strategies in RINO.

4 Parallel global router

The special-purpose hardware for parallel layout uses algorithms that were fixed in hardware, and suffers from the inability to change or to tune those algorithms. In addition, a given architecture is more economic if it is general enough to be used in a range of applications. For these reasons, all approaches discussed in this paper assume general purpose MIMD multiprocessor and distributed processors.

In essence, the routing algorithm comprises four processes: ConcurrentRouter, SequentialRouter, EnumerativeRouter and the master control process called ControlRouter. The design of the parallel routing algorithm is based on the four processes. Some important variables used in the algorithm are defined as follows:

Variables	Definition
$E_{cg}(\theta)$	The set of congested GRG edges
$N_{cg}(\theta)$	The set of congested nets
N_{open}	Random subset of $N_{cg}(\theta)$
C_{max}	Highest congestion degree among all the edges
$C_{overflow}(\alpha)$	The total overflows of GRG edges
$C_{so}(\alpha)$	The total overflows of GRG edges with weights

The parallel routing algorithm is introduced in detail as follows:

4.1 Master control process ControlRouter

```
PROC ControlRouter
FOR i= 1 TO K0 DO
    CALL ConcurrentRouter(i) store result
    as Ugood;
    CALL SequentialRouter(i) to improve
    Ugood;
    IF Ecg(1.0)= ΦTHEN RETURN
ENDFOR
IF Ecg(1.0) ≠ΦTHEN CALL EnumerativeR-
outer;
ENDPROC
```

The master control process directs the direction of the algorithm according to current routing results. The master control process does not take much time in routing large circuits. On the other hand, if we modify the control process to keep a number of current routing results, there will be a considerable alteration on the routing algorithm. Therefore, we keep the ControlRouter unchanged in the parallel algorithm.

4.2 Concurrent routing process ConcurrentRouter

```
PROC ConcurrentRouter(j)
```

```
FOR i= 1 TO K1 DO
    Suppose N_open to be random subset of
    N_cg(θ_{i,j});
    Concurrent re-route nets in N_open;
    Update U_good and U_best according to current
    solutions;
    IF E_cg(1.0) = ΦTHEN RETURN
ENDFOR
ENDPROC
```

ConcurrentRouter consists of three main operations: (1) building N_{open}; (2) concurrent re-routing of nets in N_{open}; (3) updating U_{good} and U_{best} according to routing results. Among them, the last one is a frequent memory-accessing operation, which is not useful to parallelization. Therefore, it remains serial in our algorithm.

(1) Building N_{open}. The time taken by different nets to judge congestion is approximately the same (the probability of congested edges is independent of the net size). And the number of nets in a given global routing problem is a constant. Thus, we can assign this task to each processor before routing. Further, since N_{open} is shared among all the processors, which may lead to memory-accessing conflicts, we mark each net and use one processor to put all the marked nets into N_{open}. Thus, N_{open} is built up. We divide all the nets into p groups by net-ID MOD p, each processor dealing with a group and one processor summing up all the information. Since the discrepancy in size of each group is not more than one, static load balance is obtained in this parallel operation.

(2) Concurrent re-routing of nets in N_{open}. This operation consumes the longest time in the serial algorithm and is repeated with the highest frequency, which represents a great restriction to the performance of the serial algorithm. Therefore, we focus on adding parallelism to this operation to gain a considerable improvement in parallelism of the whole algorithm. Noting that it contains complicated operations in re-routing a net, this task is assigned in dynamic. N_{open} becomes the task list of re-routing spontaneously. Each processor takes one net out of N_{open} (this operation must be mutually exclusive for N_{open} is shared), then re-route it according to current weights of GRG edges until N_{open} is empty. Since the weights of GRG edges remain unchanged throughout re-routing, accessing GRG edges need not to be mutually exclusive.

4.3 Sequential random routing process SequentialRouter

```
PROC SequentialRouter(j)
Load U_good;
Build N_cg(θ_j);
FOR i= 1 TO K2 DO
    Re-route nets in N_cg(θ_j) randomly;
    Update U_best according to current solu-
    tions;
    IF E_cg(1.0) = ΦTHEN RETURN
ENDFOR
ENDPROC
```

The essential difference between SequentialRouter and ConcurrentRouter is that nets in SequentialRouter are re-routed one by one, after which the weights of GRG edges are updated according to the re-routed net. This means that the SequentialRouter can not be paralleled in essence. Nevertheless, SequentialRouter occupies long CPU time throughout the routing process. It will add to the non-parallel time, i. e., the overhead, and will limit speed-up. Therefore, we manage to use pseudo-serial re-routing in the first several iterations of our algorithm. The re-routing of nets in a processor is serial while all the processors are parallel to improve the routing speed. The later iterations are still serial to guarantee the precision of routing solutions.

For the same reason as mentioned above in ConcurrentRouter, we dynamically assign tasks to processors in pseudo-serial re-routing. Each processor takes one net from $N_{cg}(\theta_j)$, and update the weights of GRG edges to show the increase in routing resource after remove of a routed net (this operation must be mutually exclusive for $N_{cg}(\theta_j)$ and GRG are shared). Then it re-routes the net accord-

ing to current weights of GRG edges. Re-routed nets are put back to GRG with updating in weights of GRG edges. These operations are repeated until $N_{cg}(\theta_f)$ is empty.

Since the updating of weights of GRG edges is mutually exclusive, it is obvious that the weights of GRG edges are accurate after re-routing all nets. However, other processors may update the weights of GRG edges when a processor is re-routing. And it will cause memory-accessing conflicts, which may lead to variance in calculations of congestion degree. But this problem is not a mortal wound to the algorithm since it is in the beginning of the serial iterations. The inaccuracy is local, which only influences a few GRG edges. And the error of each GRG edge is not more than p. The pseudo-serial re-routing at the beginning of iterations of SequentialRouter is to find good solutions around the current ones and to probe the potential optimization.

4.4 Global information of GRG edges

Global information of GRG edges, including C_{max}, C_s, $C_{overflow}(\alpha)$ and $C_{so}(\alpha)$, will be updated after every iteration of ConcurrentRouter and SequentialRouter. It is a frequent and time-consuming operation, which therefore needs to be parallelized. Since the time taken to deal with an edge is a fairly small constant, and the number of GRG edges in a given problem is fixed, we could assign the edges to processors in advance. We divide all the GRG edges into p groups by edge-ID MOD p, each processor dealing with a group and one processor summing up all the information. Since the discrepancy in size of each group is not more than one, static load balance is obtained in this parallel operation.

4.5 Enumerative optimizing process EnumerativeRouter

```
PROC EnumerativeRouter
Load U_best;
FOR i= 1 TO K_3 DO
    IF N_cg(1.0)= φ THEN RETURN
     Suppose N_sub to be subset of N_cg(1.0)
     satifying following:
     (1) |N_sub| < N_size.
     (2) product of routing tree numbers of
     nets in N_sub is not more than M_limit;
     (3) each net in N_sub should have at least 1
     common edge with others;
     Enumerate all the steiner tree combinations of N_sub to minish C_so(1.0);
ENDFOR
ENDPROC
```

In EnumerativeRouter we enumerate all the possible combinations of routing trees for some nets. The enumeration process can be paralleled because evaluations of all the combinations are theoretically independent. However, in realization of the algorithm, all the combinations are usually generated by recursion. There is a certain extent of order among them. At the same time, evaluation of each combination depends on the way that the combination is generated. Therefore, it is hard to realize parallel generation of combinations and their evaluations. Our approach makes the parallel of the branches operation. We try to find the net whose routing tree number K most satisfies p in N_{sub}. Then, we divide all its routing trees into p groups as equally as possible. Each processor generates and evaluates all the possible combinations of routing tree in a group. We allocate independent memory for each processor to store the weights of GRG edges, the updating of whom are guaranteed to be parallel. Finally, a processor selects the best solution from p optimal choices to be the final solution.

5 Experimental results and analysis

We implemented our parallel algorithm by using C language and tested the implementations on a SUN Enterprise 450 workstation with 4 CPUs. Five MCNC benchmark circuits and three industrial circuits have been tested. Each circuit has been tested for 12 times.

5.1 Important characteristics of test circuits

Table 1 lists some important characteristics of these 8 benchmark circuits.

Table 1 Circuit data

Test circuits	Number of nets	Number of girds
MCNC C2	745	9×11
MCNC C5	1763	16×18
MCNC C7	2356	16×18
MCNC s13207	4953	24×26
MCNC avq	21851	65×67
u05	36451	205×205
u08	54740	251×251
u28	181930	458×458

5.2 Comparison of serial and parallel algorithms

Table 2 compares the total run-times of serial and parallel algorithm and lists the speedup of parallel algorithm on multiprocessor. To large scale test circuits, the parallel algorithm can greatly reduce routing run time and obtain a speedup (I/O time of files not included) of more than 3 (the efficiency is above 75%).

Table 2 Speedup results of parallel algorithm

Test circuits	Serial/s	Parallel/s	Speed-up/s
C2	1.28	1.08	1.19
C5	3.05	1.98	1.54
C7	4.59	2.58	1.78
s13207	21.60	9.08	2.38
avq	50.79	23.66	2.15
u05	1701.94	523.43	3.25
u08	3242.49	988.99	3.28
u28	> 39600	10800.00	—

We also have approaches of timing-driven model on multiprocessor and wire length model on distributed processors for parallel global routing. The timing-driven model shows good speedups for large-scale circuits while it is not as desirable for small-scale circuit due to the non-parallelism in computation of the delay. On distributed processors, work accomplished hitherto has shown potential in providing more significant improvement in speed.

6 Conclusion

In this paper, we have presented an efficient parallel global routing algorithm. The parallel algorithm greatly reduces the routing run-time. To large scale circuits, implementation on 4 processors can obtain a speedup of above 3 with no degradation of routing quality. Meanwhile, this parallel routing algorithm makes full use of the advanced routing strategies.

References

[1] Hong Xianlong, Yan Xiaolang, Qiao Changge. VLSI layout theories and algorithms. Beijing: Science Press, 1998 (in Chinese) [洪先龙, 严晓浪, 乔长阁. 超大规模集成电路布图理论与算法. 北京: 科学出版社, 1998]

[2] Tang Pushan. Theories and approaches for VLSI computer aided design. Shanghai: Fudan University Press, 1990 (in Chinese) [唐璞山. VLSI 计算机辅助设计理论和方法. 上海: 复旦大学出版社, 1990]

[3] Jing Tong, Hong Xianlong, Cai Yici, et al. On the key technologies of performance-driven global routing. Chinese Journal of Software, 2001, 12(5): 677(in Chinese) [经彤, 洪先龙, 蔡懿慈, 等. 性能驱动总体布线的关键技术及研究进展. 软件学报, 2001, 12(5): 677]

[4] Olukotun O A, Mudge T N. A preliminary investigaton into parallel routing on a hypercube. Proc Design Automation Conf, June 1987: 814

[5] Martonosi M, Gupta A. Tradeoffs in message passing and shared memory implementations of a standard cell router. Proc Int Conf Parallel Processing (ICPP89), 1989: 88

[6] Rose J. Locusroute: a parallel global router for standard cells. Proc of 25th DAC, 1988: 189

[7] Rose J. Parallel global routing for standard cells. IEEE Trans Computer Aided Design Integrated Circuits Systems. 1990: 1085

[8] Sechen C, Sangiovanni-Vincentelli A. TimberWolf placement and routing package. IEEE Journal of Solid-State Circuits, 1985, 20(2): 432

[9] Vecchi M P, Kirkpatrick S. Global wiring by simulated annealing. IEEE Transaction on CAD, 1983, 2(4): 215

[10] Kirkpatrick S, et al. Optimization by simulated annealing. Science, 1983, 220: 671

[11] Brouwer R J, Banerjee P. PHIGURE: a parallel hierarchical global router. Proc 27th Design Automation Conf, June 1900: 360

[12] Chiang C, Sarrafzadeh M. Global routing based on steiner min-max tree. IEEE Transaction on CAD, 1990, 9(12): 1318

[13] Chiang C, Sarrafzadeh M, Wong C K. A powerful global router: based on steiner min-max trees. Proc ICCAD, 1989: 2

[14] Ting B S, et al. Routing techniques for gate array. IEEE Transaction on CAD, 1983, 2(4): 310

[15] Lee K, Sechen C. A new global router for row-based layout. Proc ICCAD, 1988: 180

[16] Heisterman J, Lengauer T. The efficient solution of integer programs for hierarchical global routing. IEEE Transaction on CAD, 1991, 10(6): 748

[17] Burstern M S, Pelavin P. Hierarchical wire routing. IEEE Transaction on CAD, 1984, 3(3): 250

[18] Parng T M, Tsay R S. A new approach to sea-of-gate global routing. Proc ICCAD, 1989: 52

[19] Chen Zhichao, Bo Jianguo, Ma Zuocheng, et al. Algorithm on superplane routing. Chinese Journal of Semiconductors, 1997, 18(2): 128(in Chinese) [陈志超, 薄建国, 马佐成, 等. 超平面概略布线算法的研究. 半导体学报, 1997, 18(2): 128]

[20] Zhuang Changwen, Fan Mingyu, Li Chunhui, et al. Ant colony switchbox router based on coordination mechanism. Chinese Journal of Semiconductors, 1999, 20(5): 400(in Chinese) [庄昌文, 范明钰, 李春晖, 等. 基于协同工作方式的一种蚁群布线系统. 半导体学报, 1999, 20(5): 400]

[21] Yang Changling, Yan Xiaolang. SERR: a simulated evolution performance-driven global router. Chinese Journal of Semiconductors, 1998, 19(2): 151(in Chinese) [杨昌玲, 严晓浪. SERR: 基于模拟进化技术的性能驱动总体布线算法. 半导体学报, 1998, 19(2): 151]

[22] Li Zhiyan, Yan Xiaolang. High speed multilevel staged clock routing. Chinese Journal of Semiconductors, 2000, 21(3): 290 (in Chinese) [李芝燕, 严晓浪. 高速多级时钟网布线. 半导体学报, 2000, 21(3): 290]

[23] Ting B S, Tien B N. Routing techniques for gate array. IEEE Transaction on CAD, 1983, 2(4): 301

[24] Bao Haiyun, Jing Tong, Hong Xianlong, et al. A novel random global routing algorithm independent of net ordering. Chinese Journal Computers, 2001, 24(6): 574(in Chinese) [鲍海云, 经彤, 洪先龙, 等. 一种新的与线网顺序无关的随机优化总体布线算法. 计算机学报, 2001, 24(6): 574]

[25] Carden R C Ⅳ. Microelectronics global routing and feasibility estimation using a multicommodity flow approach. Doctor Dissertation. Department of Computer Science, University of California, San Diego, 1991

一个快速高效进行布线拥挤优化的总体布线器*

许静宇[1] 鲍海云[1] 洪先龙[1] 蔡懿慈[1] 经 彤[1] 顾 钧[2]

(1 清华大学计算机科学与技术系, 北京 100084)

(2 香港科技大学计算机科学系, 香港)

摘要: 设计实现了一个高效的线长模式下基于多处理机的并行总体布线器. 通过对非时延驱动模式下串、并行算法的总运行时间和求解时间的比较, 表明该并行算法能够在保证求解质量无明显变化的前提下大大加快总体布线算法的求解速度. 同时, 也提出了基于分布式体系结构的并行总体布线算法.

关键词: 总体布线; 布线拥挤优化; 总体布线图(GRG); 并行算法

CCACC: 7410D

中图分类号: TN47 **文献标识码**: A **文章编号**: 0253-4177(2002)02-0136-07

* 国家自然科学基金(No. 60076016), 国家 973 重点基础研究发展规划(No. G-1998030403) 和中国博士后科学基金(No. [2000]23) 资助项目

许静宇 1976 年出生, 博士研究生, 主要从事总体布线的研究工作.

鲍海云 1972 年出生, 博士研究生, 主要从事总体布线的研究工作.

洪先龙 1940 年出生, 教授, 主要从事布图的研究工作.

2000-06-19 收到, 2001-09-28 定稿