PACKER: A SWITCHBOX ROUTER BASED ON CONFLICT ELIMINATION BY LOCAL TRANSFORMATIONS*

Sabih H. Gerez and Otto E. Herrmann
University of Twente, Faculty of Electrical Engineering
Laboratory for Network Theory
P.O.B. 217, 7500 AE Enschede, The Netherlands

Abstract

PACKER is an algorithm for switchbox routing, based on a novel approach. In an initial phase, the connectivity of each net is established without taking the other nets into account. In general this will give rise to conflicts (short circuits). In the second stage the conflicts are removed iteratively using *connectivity preserving local transformations*. They "reshape" a net by displacing one of its segments without disconnecting it from the net. The transformations are applied in a systematic way using a scan line technique. The results obtained by PACKER are very positive: it solves all well-known "benchmark" examples.

1 Introduction

This paper explains the PACKER algorithm for switchbox routing. It uses a routing model with the following properties:

1. routing is performed on an orthogonal grid;
2. two layers of wiring are used;
3. horizontal and vertical connections are allowed in both layers;
4. all terminals have a fixed position and the layer of a terminal is fixed, i.e. a terminal has to be extended in this layer for at least one grid unit into the switchbox. Not all the terminals on the same side need to enter the switchbox in the same layer.

PACKER solves switchbox routing problems in two steps. First it establishes the connectivity of the nets by connecting the terminals of each net in a straightforward way, without taking the other nets into account. The configuration arrived at in this way is likely to contain many conflicts (short circuits). In the second stage the algorithm tries to remove existing conflicts by "reshaping" the nets without destroying their connectivity. This is done my means of *Connectivity Preserving Local Transformations* (CPLTs). Each transformation moves one segment one grid unit aside. A systematic approach is used for the reshaping: a scan line is moved in one of the four directions (from left to right, right to left, top to bottom, bottom to left) and as much as possible of the line segments located on the scan line are packed in the two layers on that position. The rest of the segments, incompatible with those selected to remain, are moved to the next position using CPLTs. This type of scanning is repeated in different directions until either a solution is found or the algorithm fails to find a solution within a maximum number of scans given by the user.

*The research, the results of which are presented here, is part of the project "Innovative CAD Tools for IC Design", supported by the Foundation FOM (TEL 330408).

2 Relation to Other Work

Most of the successful routers reported recently [3,4,8,11,14,16] generate intermediate configurations which are free of conflicts at the expense of incompletely connected nets. In PACKER, as well as in some algorithms that were developed independently at around the same time [9] (and also [15], but less strictly), the nets are always connected at the expense of conflicts. The idea of allowing conflicts in intermediate configurations is not new. Rubin, in an algorithm for printed circuit boards [13], uses the information on conflicts to update cost factors in a Lee-type router. The same idea also can be found in more recent algorithms by Rosenberg [12] and Linsker [10]. Lin, Hsu and Tsai [9] add a probabilistic element to a similar approach and succeed in applying the method to switchbox routing.

The use of what has here been called "connectivity preserving local transformations" is mentioned in [2], in an application for global routing, calling them "parallel displacements of the straight sections". No details are given, however, on *how* to get rid of overcongested channels (the counterpart of a conflict in global routing) in a systematic way. The "weak modifications" of [14] and the "wire pushing operations" of [15] have some similarities with CPLTs. In both cases, however, they are not the only methods for the modification of the wiring: Lee-routing, rip-up and reroute are also used. PACKER does not rely on anything else but the CPLTs.

The work which comes closest in philosophy to the ideas behind PACKER is perhaps the one reported by Fisher [5]: it also tries to remove conflicts iteratively by means of transformations. However, the strategy for conflict resolution has strongly been influenced by the peculiarities of PCB routing.

3 Connectivity Preserving Local Transformations

In PACKER the routing is represented by *line segments*, whose lengths are a multiple of the grid unit, and dimensionless *nodes* that interconnect them. The *layer* is an attribute of the line segment. This data structure turns out to be appropriate for the implementation of the connectivity preserving local transformations.

CPLTs are the elementary moves in the reshaping of nets. Two different types of CPLTs exist:

1. moving a line segment a single grid position aside;
2. splitting a line segment longer than one grid unit into two line segments.

Figure 1 shows some simple transformations of both types. When a segment is moved, the principle used is that as least as

ISCAS '89

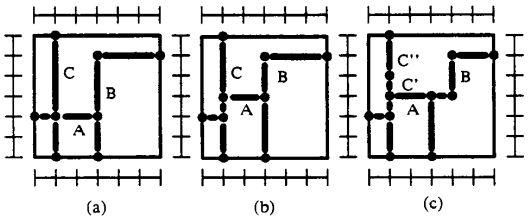

Figure 1: Some simple moves: starting from (a), moving segment A gives (b) and moving B together with the splitting of C gives (c).

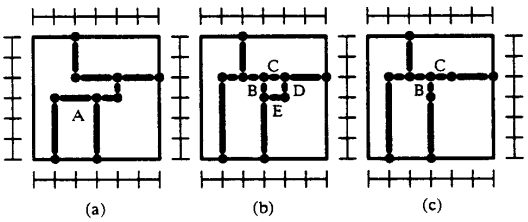

Figure 2: Moving segment A in configuration (a) creates the cycle B-C-D-E as depicted in (b); D and E are removed, forming the longest subsequence in the cycle, that can be discarded legally (c);

possible of the routing of the net should be affected. In simple cases only the perpendicular segments at the end points of the segment to be moved are made shorter or longer, they are split or newly created to maintain connectivity. The situation becomes more complicated when the destination area already contains segments of the same net. In such a case the segment to be moved has to be *merged* with those already present. The most complicated situation occurs when merging creates a cycle. In a cycle there is at least one segment that can be removed without losing the connectivity of the net. PACKER removes any of the longest sequences of segments in the cycle that do not disconnect the net, see Figure 2.

4 Control of the Transformations

The starting configuration of the switchbox is obtained in the initial routing phase. In this phase for each net all its terminals are interconnected. The method of interconnection is not very essential, as most nets will be modified anyhow; the results reported in this paper, have been obtained using a *quasi-minimal Steiner tree* method, in which a minimal *spanning tree* (Prim's algorithm, see e.g. [7]) is used to determine which pairs of points have to be interconnected. When the net has more than 2 terminals, the L-shaped interconnections between two points are chosen in such a way that they exploit as much as possible of the wiring already laid out.

The CPLTs are performed on the segments in the layout during a *scanning* process. A scan line is moved from one extreme to the other in one of the four directions: from left to right (LR), right to left (RL), top to bottom (TB) and bottom to top (BT). At each position, the set of segments along the scan line, S, is inspected, while the, normally false, assumption is made that the segments perpendicular to the scan line have the correct location. From S a maximal subset of compatible segments, SU, is selected and its segments are given a position in either of the two layers without creating conflicts. The segments in $S \setminus SU$ are incompatible with those in SU and are moved to the next position in the scanning direction using CPLTs. Weights are assigned to all segments in S, in order to guide the computation of SU. The weight of a segment is the summation of numerical values related to the following: the length of the segment, the presence of constraints (some kind of local vertical constraints, see [6]), the presence of segments of the same net ahead in the scanning direction (such that the two segments can be merged or that jogs are removed) and an effect that privileges different nets in different moments of the computation to prevent looping. Actually, the computation of SU from S is a weighted variant of the *maximal independent set* problem [7].

Iteration no.	Sequence						
i	TB	RL	BT	LR			
$i+1$		RL	BT	LR	TB		
$i+2$			BT	LR	TB	RL	
$i+3$				LR	TB	RL	BT

Table 1: The iteration scheme.

In general a single scan through the switchbox will not be enough to resolve all the conflicts and all the wire segments which could not be given a place, will get accumulated at a row or column next to the switchbox boundary. A scan in another direction will have to solve the remaining conflicts.

The top level control of PACKER consists of a number of *iterations*, each iteration calling the scanning routine for each of the four directions in some order. An iteration has the following property: if the layout is not modified during the four calls, one is sure that there are no conflicts left and that a solution has been found. (A single scan ignores conflicts in the perpendicular segments and does not inspect the last row or column.) PACKER uses an iteration scheme in which the sequence of calls is shifted cyclically with respect to the previous one, see Table 1.

After generating the initial routing, PACKER performs a number of iterations on the problem using the iteration scheme of Table 1. As soon as the configuration is not modified during an iteration (of four scans), the algorithm stops, reporting success. At that moment a simple via optimization heuristic is executed: it removes redundant vias in most cases, but not in all. When no solution is found within the number of iterations given by the user, the algorithm stops and reports failure.

The worst case time complexity of the algorithm is $\mathcal{O}(sr^4)$ [6], where s is the number of scans and r is the maximal height/width of the switchbox.

5 Results

PACKER has been implemented in Common Lisp. All the timing results given refer to compiled code on an Apollo DN 4000 workstation with 8 Mb of main memory.

In this section the results obtained by PACKER for a number of well-known switchbox routing problems will be presented and compared with the results of other successful routers. When comparing the quality of the results, it should be kept in mind that the minimization of total wire length and number of vias nowhere appears as an explicit part of the algorithm in PACKER.

example's name	iter-ations	run time *sec*	modifi-cations	wire length	vias
difficult	14	56	2663	565	46
more difficult	70	1400	24339	552	43
terminal intensive	28	210	3393	630	53
augmented dense	5	31	414	529	32
modified dense	6	36	391	510	30
pedagogical	14	91	1790	407	46
comp-1	14	160	3161	528	45

Table 2: The results obtained by PACKER for a number of benchmark switchbox routing problems.

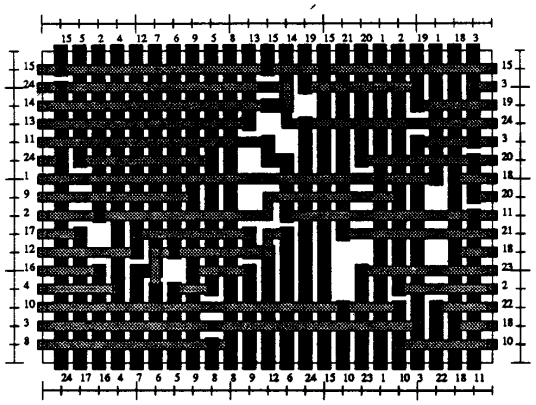

Figure 3: A solution found by PACKER for the terminal intensive switchbox.

PACKER simply stops when it finds a solution without conflicts. Besides, the results presented here are not the best results ever found by PACKER, but those obtained with a fixed choice of the parameters related to the weights. This can be motivated by stating that the user in practice should not be bothered with the peculiar parameter of each tool that he/she is using, sometimes at the expense of not being able to obtain the optimal result for each individual problem.

Table 2 gives some relevant data for each example. These data are: the number of iterations performed to find the solution, the execution time, the number of modifications (segment moves, splits and layer changes) performed to reach the solution from the initial routing, the total wire length and the number of vias. The same example names are used as in [4], with the addition of *comp-1*, a compressed variant of the *more difficult example*, which is mentioned in [15].

Some of the results have been illustrated in the Figures 3 and 4, while the comparisons with BEAVER [4], WEAVER [8], MIGHTY [14], SILK [9] and CODAR [15] have been made in the Table 3. The run time has been included in Table 3 in order to have a global idea of the performance. More cannot be done when different programming languages on different types of computers are used. In the case of PACKER a run time efficient implementation was not a primary goal. Rather, it was the intention to build a flexible tool allowing easy experimentation with a completely novel approach to switchbox routing.

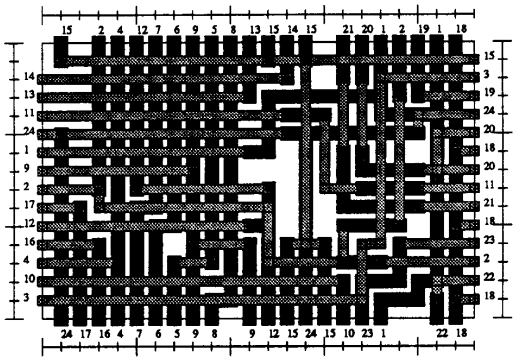

Figure 4: A solution found by PACKER for the example "comp-1".

example	router	wire length	vias	run time *sec*
difficult	WEAVER	531	41	1500
	BEAVER	547	43 (35)	1
	CODAR	544	?	15
	PACKER	565	46	56
more difficult	MIGHTY	541	39	4
	BEAVER	536	42 (34)	1
	SILK	528	36	69
	CODAR	545	?	17
	PACKER	552	43	1400
terminal intensive	WEAVER	615	49	1800
	MIGHTY	629	50	?
	BEAVER	632	53 (46)	1
	SILK	616	49	?
	CODAR	630	?	21
	PACKER	630	53	210
augmented dense	MIGHTY	530	32	?
	BEAVER	529	32 (27)	1
	CODAR	529	?	10
	PACKER	529	32	31
modified dense	WEAVER	510	29	920
	MIGHTY	510	29	?
	BEAVER	510	29 (26)	1
	SILK	510	29	?
	CODAR	510	?	11
	PACKER	510	30	36
pedagogical	BEAVER	396	38 (31)	1
	PACKER	407	46	91
comp-1	MIGTHY	fail	-	-
	CODAR	529	42	50
	PACKER	528	45	160

Table 3: The results obtained by PACKER for a number of benchmark switchbox routing problems.

As far as the comparison with BEAVER is concerned, the following should be remarked: BEAVER takes the freedom to assign a terminal to the most convenient layer with the goal of saving vias, whereas most routers assume that terminals have a fixed layer. For a fair comparison the figures for BEAVER have been corrected to match the fixed terminal specifications, giving the original ones inside parentheses.

(217) 333-4789

Further information can be obtained from Dr. W. Kenneth Jenkins.

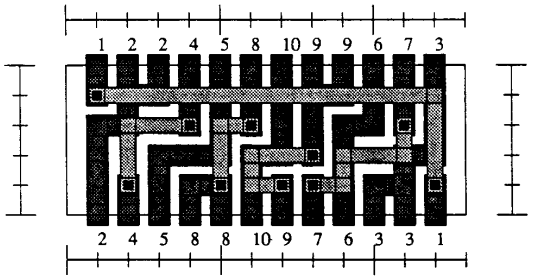

Figure 5: A solution found by PACKER for Burstein's difficult channel.

Channel routing problems that do not have floating terminals at the "open sides" can be described as switchbox routing problems, when the number of rows is fixed in advance. PACKER can solve small and medium-size channel routing problems, that have this property.

One of the most interesting results achieved by PACKER is a solution to a problem originally published in [1]. A 5-row solution required two empty columns in the middle with a routing model where each layer had its own orientation for wire segments. MIGHTY, without the restricted routing model, solved the problem using 4 rows and a single empty column in the middle. PACKER, using the same unrestricted model, is able to solve it in 4 rows, without any empty columns. The solution is given in Figure 5.

6 Final Remarks

We can conclude that a completely novel approach for switchbox routing (initial routing with conflicts followed by stepwise reshaping) has turned out to give interesting results. Many aspects of PACKER that could not be explained here can be found in [6] together with many more results.

References

[1] M. Burstein and R. Pelavin. Hierarchical channel router. *Integration, The VLSI Journal*, 1:21–38, 1983.

[2] K.A. Chen, M. Feuer, K.H. Khokhani, N. Nan, and S. Schmidt. The chip layout problem: an automatic wiring procedure. In *14th Design Automation Conference*, pages 298–302, 1977.

[3] S. Chen, M. Lin, and W. Zhuang. A new algorithm for four-side channel routing. In *Proceedings of 1985 China International Conference on Circuits and Systems*, pages 73–76, Science Press, Beijing, China, 1985.

[4] J.P. Cohoon and P.L. Heck. Beaver: a computational-geometry-based tool for switchbox routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 7(6):684–697, June 1988.

[5] R.S. Fisher. A multi-pass, multi-algorithm approach to PCB routing. In *15th Design Automation Conference*, pages 82–91, 1978.

[6] S.H. Gerez. *PACKER: A Switchbox Router Based on the Stepwise Reshaping of Fully Connected Nets.* Technical Report EL-BSC-88C146, University of Twente, Faculty of Electrical Engineering, October 1988.

[7] A. Gibbons. *Algorithmic Graph Theory.* Cambridge University Press, Cambridge, 1985.

[8] R. Joobbani and D. Siewiorek. Weaver: a knowledge-based routing expert. *IEEE Design & Test*, 3(1):12–23, February 1986.

[9] Y.L. Lin, Y.C. Hsu, and F.S. Tsai. A detailed router based on simulated evolution. In *International Conference on Computer-Aided Design*, pages 38–41, 1988.

[10] R. Linsker. An iterative-improvement penalty-function-driven wire routing system. *IBM Journal of Research and Development*, 28(5):613–624, September 1984.

[11] W.K. Luk. A greedy switch-box router. *INTEGRATION, the VLSI Journal*, 3:129–149, 1985.

[12] E. Rosenberg. A new iterative supply/demand router with rip-up capability for printed circuit boards. In *24th ACM IEEE Design Automation Conference*, pages 721–726, 1987.

[13] F. Rubin. An iterative technique for printed wire routing. In *11th Design Automation Workshop*, pages 308–313, 1974.

[14] H. Shin and A. Sangiovanni-Vincentelli. A detailed router based on incremental routing modifications: Mighty. *IEEE Transactions of Computer Aided Design of Integrated Circuits*, CAD-6(6):942–955, November 1987.

[15] P.S. Tzeng and C.H. Sequin. Codar: a congestion-directed general area router. In *International Conference on Computer-Aided Design*, pages 30–33, 1988.

[16] C.S Ying, X.L. Hong, and E.Q. Wang. Draft: an efficient area router based on global analysis. In *Proceedings International Conference on Computer-Aided Design*, pages 386–389, 1987.