# A LEGAL algorithm following global routing

Marcelo de Oliveira Johann

Glauco Borges Valim dos Santos

*Pontifícia Universidade Católica do Rio Grande do Sul. BR 472 Km 07, Campus Universitário II,*
*Cx. Postal 249, CEP 97500-970, Uruguaiana – RS, Brasil*
*Phone: (xx55) 4131515, Fax: (xx55) 4131280*
*E-mails: Johann@pucrs.campus2.br, glauco@pucrs.campus2.br*


Ricardo Augusto da Luz Reis

*Universidade Federal do Rio Grande do Sul. Av. Bento Gonçalves 9500, Campus do Vale, Bloco IV,*
*P.O.Box 15064, Cep 91501-970, Porto Alegre –, RS, Brasil.*
*Phone: (xx51) 33166830, Fax: (051) 3191576*
*E-mail: reis@inf.ufrgs.br*

## Abstract

*This papers presents an implementation of a LEGAL algorithm following a previous established global routing. LEGAL algorithms were first proposed by Johann in 1994, where the ideas from the most efficient channel routing algorithms were combined and used to process detailed routing of large areas in almost linear time. Other implementations produced good detailed results but had no means of making a global routing distribution in bigger circuits, which is a task for global routing. Now we present the first implementation of a detailed LEGAL routing that can obey global paths previously chosen for each net. The mechanisms developed to isolate global and local decisions and still keep LEGAL routing being made over the whole area are the focus of this paper. The implementation should be tested and refined further to allow optimizations and become competitive to other approaches in practice.*

## 1. Introduction

As the IC technology fabrication evolved, we passed from the old channel routing models to new multi-layer models were there are no dedicated spaces for routing. Having no dedicated spaces in the footprint means that we do not have small distinct problems, but actually a single large space to place cells and make connections, in a model that is called area routing.

Usually, the divide-and-conquer technique is used to break down the entire routing problem into a set of sub-problems that can be solved separately. When doing such a global routing step to an area routing problem, we get a set of switch-boxes without any definition about the positions on its boundaries where the nets have to cross.

Although the switch-box itself can be efficiently solved, its definition is not easily derived from the global routing. Thus, a new step was introduced for solving the so-called Cross Point Assignment problem (CPA).
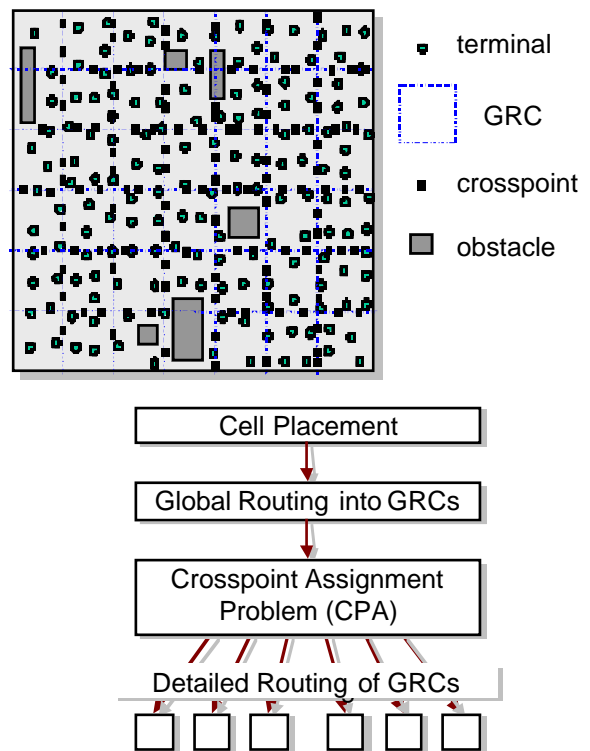


**Figure 1 – Area routing broken down into a set of GRCs with crosspoints**

As a separate problem, CPA can be solved with good heuristics, but its characteristics of global problem are clear. In fact, one approach to make this assignment is to

look at an entire row (or column) of GRCs (Global Routing Cells) and make sure net crossing is minimized before assigning an exact position for each net at each cell boundary [12]. The problem that arises on such an approach is, basically, lack of flexibility. For example, if you decided that some connection must be routed through some switchbox at some specific locations (interface), it will be mandatory that the switchbox router can route it there. On the other hand, if you are less aggressive in the promises of about what a switchbox can handle, when there is space left on it, this space cannot be used to accommodate even a single overflow from one of its neighbors. This may explain why maze routers are still the most widely used algorithms for routing, despite its problems of huge time consumption and memory usage. They are the most flexible.

On the other hand, the allocation of routing tracks by maze routers can be considered random, and we cannot expect they will appropriately use the available space. Thus, our target was to develop a new detailed routing algorithm that could avoid the problems of both approaches. It is expected to:

- run in a large area with cell pins inside;
- route all nets at the same time, avoiding huge running times and net ordering problems;
- consider constraints within this area;
- be efficient in allocating routing resources.

## 2. LEGAL Algorithm

LEGAL is a generalization of two of the most important channel routing techniques, Left-Edge [3] and Greedy [14] algorithms. We observed that Greedy and Left-Edge are complementary routing techniques, and that it is possible to combine them and build a new efficient area routing algorithm, called LEGAL. LEGAL processes a symbolic view of the circuit line by line, having as its main control a sequence of steps as in the Greedy algorithm. At each line and step, there are segments that are needed or at least useful for solving the problem. Those that could be implemented are selected in a Left-Edge manner. Fig. 2 shows a skeleton of the LEGAL algorithm.

While the horizontal scanning considers the segments in a simple Left-Edge way, the priority in which they must be considered is embedded in the Greedy-like steps. The meaning of this processing is as follows. While each line of the circuit is being routed, previous dependencies are solved, since they are required, and the rest of the space is used to do what can be done in advance. This is a simplified view of the existing priorities. The algorithm is greedy in nature, but is thought to take only local decisions according to clear priorities, and the fact that it considers all nets at the same time let it potentially better converge into a solution with 100% routing completion. One should observe that, while shortest path search

methods like maze routers are known to be exact, the overall approach of making a single connection at a time is also a greedy technique.

Each step performs a specific task: 1- add new pins; 2- join nets present at more than one column (split); 3- approximate split nets; 4- move nets closer to their next terminal. There are separate functions to identify each of these conditions as well as to make the horizontal scanning to find positions that can be used for that specific connection. Fig. 2 shows only the step functions for step 1 and 2. Finally, legal_move and legal_union are utility functions that process horizontal movements and unions of split nets, updating column information and generating the corresponding routing.

```
Legal  algorithm:

for each net pin
     put pin in Terminal[pin.y];
for (line=0; line<circuit.y; ++line)
  {
  extend live columns;
  step1(line); // add new pins
  step2(line); // join split nets
  step3(line); // approximate split nets
  step4(line); // move nets closer to target
  }
```

```
legal_step1:

for each pin in Terminal[line]
  {
  calculate target_x; // Left or Right
  found=scan_step1(pin.x,target_x,&found);
  if (not found)
    error("Cannot includ new pin");
  if (Column[found] has no net)
    legal_move(pin.x,found);
  else
    legal_union(pin.x,found);
  }
```

```
legal_step2:

for x = 0 to circuit.x
  if Column[x] has a net
    if the net at Column[x] is split
      {
      found=scan_step1(pin.x,
        target_x,&found);
      if (found)
        legal_union(pin.x,found);
      }
```

**Figure 2 – Structure of a LEGAL algorithm**

In the greedy channel algorithm, the target for a connection can be the top or the bottom of the channel, or even to stay at the same column when there are terminals of the same net on both channel sides. However, in an area routing, the targets for each net segment could not be

modeled with a so simple criteria. Since we still have to know exactly where to bring each connection in order to the algorithm to operate, we had two options: consider a previous global routing step providing the management of the targets for each connection, or to use an internal global planner, dynamically calculated across the routing. The last option was considered as a starting point, mainly because it is closer to the detailed information the router deals with. It is necessary anyway even with a global router.

## 2.1. Using LEGAL for Detailed Routing Under Constraints

The first LEGAL implementation was tried in 1994 as an alternative for maze routers in the MARTE routing system [Joh94] [Joh95]. MARTE tool was developed to make detailed routing over partially transparent cells, and thus had to consider constraints to metal layers and via holes. The LEGAL implementation did not succeed for larger circuits mainly due to lack of support from MARTE data structures, which were not sufficiently robust to identify original multi-point terminals and groups of terminals already connected. However, it did prove that LEGAL algorithms work and make detailed routing in linear time for small circuits, even considering area constrains for the first metal layer.

## 2.2. Using the LEGAL Approach for Global Routing

Perhaps the most successful implementation of a LEGAL algorithm in practice was for global routing, instead. GAROTA [4] [5] is a routing system specially developed for a particular gate array architecture, as part of a cooperation between UFRGS (Universidade Federal do Rio Grande do Sul) and CTI (Centro de Tecnologia para Informática) [1], and in the last version a previous simple global routing algorithm was substituted by an adaptation of LEGAL. This implementation has solved all previous global routing problems (which were channels with tracks overflow) while keeping the fast running times that characterizes GAROTA. LEGAL is being currently used in the Beta 2.5 version of the system, which can completely route all circuits tried in less that 5 seconds of CPU (in an old Pentium 100MHz) for 12.500 transistors master-slices.

## 2.3. A Simple and Generic Implementation

As we look for a better specification of this kind of algorithm, we developed a very simple implementation restricted to only point-to-point connections and considering no area constraints [10]. This implementation, called **ilegal**, has 623 lines and was developed in only one week. The table below shows running times and routing completion with maze routers (MARTE) and the **ilegal** program. Both systems are

realizing only point-to-point connections in these experiments, which are ISCAS benchmark circuits placed into a sea-of-cells master-slice and decomposed into 2 pin nets. We observe that for small circuits the routing power is quite similar to very optimized maze algorithms, yet using little CPU time.

**Table 1 - Comparing *ilegal* against an optimized maze router.**

| circuit | | optimized MARTE | | ilegal | |
|---|---|---|---|---|---|
| name | nets | CPU (s) | % rot | CPU (s) | % rot |
| s28b | 79 | 6.0 | 98.73 | 0.0 | 98.73 |
| s386b | 546 | 26.0 | 100 | 2.0 | 100 |
| s510b | 590 | 48.0 | 99.66 | 3.0 | 97.79 |
| c880b | 819 | 76.0 | 99.87 | 4.0 | 99.75 |
| c499b | 990 | 68.0 | 99.89 | 6.0 | 100 |
| s1494b | 1870 | 425.0 | 98.12 | 12.0 | 89.51 |

Fig. 3 shows the layouts for the c499 circuit obtained by **ilegal** (at left) and MARTE´s (at right) algorithms, were we can observe that the routing distribution is not as good as distribution of the random generated paths of maze routers. The lack of global routing does not allow the **ilegal** implementation to have good routing distribution for bigger circuits, like s1494, where it starts to fail in completing all the routing. We can observe that even for the small c499 circuit, maze routers present a better homogeneity in the resulting net distribution. At that moment **ilegal** tended to produce some congested regions as it tries to move nets closer to their destination without any consideration on congestion. All these movements should be avoided.
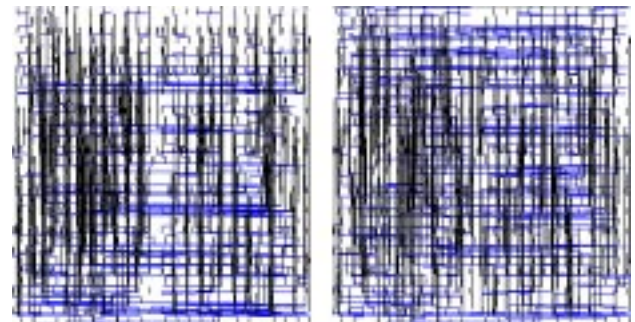


**Figure 3 – Layouts of c499b by *ilegal* (left) and MARTE (right)**

The interaction with a global router is exactly what was missing. Improving local decisions is of course another possibility, but it will be used just to optimize the routing and get cleaner connections, as it is not able to globally distribute them as global routing does. Before that, however, we need to clearly separate local and global decisions in an implementation that considers multi-pin connections.

## 3. LEGAL Following Global Routing

### 3.1. Global routing information

The first question that comes to mind is how the global routing is defined. Our response to this question is that the global routing of a net is a rectilinear graph over the global routing cells (GRCs), covering all pin locations, and using additional cells where we have turns or multiple connections (thus, a Steiner Tree). This structure represents a multi-pin network with only node-to-node connections. Some nodes are **global cell pins**, others are **virtual pins** serving as references only, and a third class of nodes is composed of the ones representing **local pins**. Local pins do not have global routing, and need only to be locally connected to a global pin. An example of a textual representation of it is shown in Fig. 4. Algorithms to get this representation from any global routing result are very straightforward, but chances are that you already have this information stored in a similar way.

```
area(xsize, ysize, globalx, globaly);
net(name, num_of_pins, [pins], [con])
pin(num, x, y, gx, gy, type)
con(pin1_number, pin2_number);
area(70,70,7,7);
net (escada,6,
     pin(1,8,8,1,1,global),
     pin(2,14,8,2,1,virtual),
     pin(3,14,14,2,2,global),
     pin(4,14,28,2,3,global),
     pin(5,28,28,3,3,virtual),
     pin(6,28,35,3,4,global),
     con(1,2),
     con(3,2),
     con(3,4),
     con(5,4),
     con(5,6) );
net (estrela, 5,
     pin(1,14,30,2,3,virtual),
     pin(2,5,31,1,3,global),
     pin(3,14,17,2,2,global),
     pin(4,29,28,3,3,global),
     pin(5,15,45,2,5,global),
     con(1,2),
     con(1,3),
     con(1,4),
     con(1,5) );
```

**Figure 4 – Example of global routing definition**

You must have observed that virtual pins do have exact positions assigned to them. This is not a simple

task. In fact, after some discussions, it was chosen to first implement the detailed LEGAL routing using this final information, but there is no simple algorithm to optimally assign these positions, and it can be viewed as our version of cross-point assignment. However, it is easy to see that it represents much less overhead and lack of flexibility than the CPA itself. Consider the simple net of Fig. 5. Using the divide-and-conquer with CPA, you have a total of 4 positions to assign instead of only one in our approach.
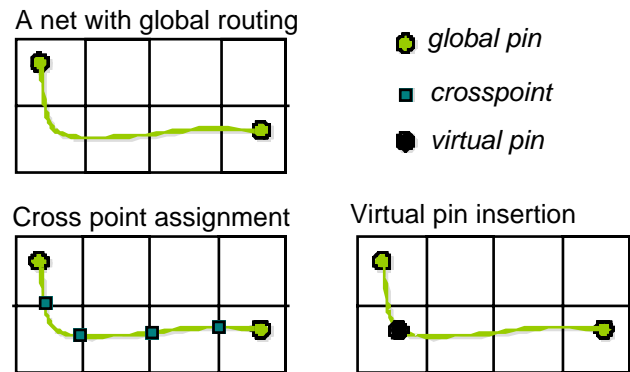


**Figure 5 – How virtual pins are better than cross point definitions**

The same happens in other practical examples. A possible solution is, for each given pin, to look at his neighbors and copy their coordinates. This algorithm for assignment only fails when we have virtual pins that are neighbors. Another possibility is to let the LEGAL assign positions on the fly, what complicates a lot the algorithm code. For now, we will keep the exact pin information as an input to demonstrate LEGAL's ability to follow global routing.

### 3.2. Network of multiple pins

Networks of multiple pins were always a concern to implement on LEGAL algorithms. There are several implications of their existence. First, the destination of each one is not clear. It depends on the pins that are close, and also on the global planning for the net. A connection cannot be routed going to the left and right alternately to be close to its targets. As we will see, global routing comes to help on this question.

Second, if a net is present at several columns in a given line, how this information is kept, and how can we identify which net segments we want to be connected and which we do not?

Third, if a given operation is performed on a network in a given line, how to select in which columns the network will remain in routing to hit further terminals?

All these questions are partially local and partially global decisions, and then comes the biggest problem: how to isolate local decisions from global decisions, following global planning while making all detailed routing at the same time, if they look all the same? The

answers derived from the global routing specification, using a new concept called **half**.

### 3.3. Half: independent routing parts

A half is **half a connection**, thus the name. Each node-to-node connection in the global routing graph is interpreted in the LEGAL code as two halves, one coming from one node and the other from its neighbor. It is easy to see that as far as we consider only global and virtual pins, each pin may have at most 4 halves originating on it. The trick here is to make halves completely **independent** and **transparent** to each other while they are from the same net. Independence means that each half has its own destination, which is the other half, its soul mate, and will not mind on what other halves are doing. On the other hand, transparency means that halves from the same net can occupy the same column if they want to, and can make movements (moves or unions) over positions already occupied by its network.

These definitions have several implications to an implementation, and fortunately all of them give us simplicity. To insert a new pin into routing (greedy step 1), it is sufficient to find a free position as close to the terminal as possible, and put all the halves on it. Further movements of halves that want to go in different directions will be completely independent on each other, and will not block on this terminal or the movement already made, since they know it is from the same net.

Other greedy steps are exclusively half based. In step 2, for example, where split nets have to be united, we have just to identify a pair of soul mate halves that want to join together. The operation then unlink these halves form their positions and eliminate both from the routing, making a single horizontal connection that represents their union. Other steps that approximate halves from each other are executed only when these halves exist, and this mechanism completely implements all the global routing intentions.

As a very simple example, observe the net of Fig. 5. It had only two global pins, but the intermediate virtual pin was inserted to specify global routing. Then, each one of the four resulting halves has a very straightforward target in mind. The algorithm will not make any global decisions of taking other paths to connect end terminals A and B, since they do not look into each other, but will actually decide only the exact routes that these connections may take according to the other networks that are being routed in the same space at the same time.
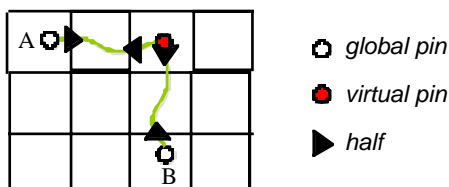


**Figure 5 – Virtual pins isolate the destinations**

The reader would have to think a bit to check that these entities are simple to implement. Routing columns have to store the only net number and a linked list of halves. Each half has a pointer to its soul mate (target half), its net number and current $x$ coordinate (column). This information can be further used to accelerate identification testes on each step, which are now based on searching the linked lists and comparing a given half to other halves at next positions. Note that searching these lists is not a computationally hard task, since they are empty or have a limited number of halves, typically ranging from 1 to 4.

### 3.4. Dealing with local pins

Local pins, which were not considered at the first moment, can be dealt with by inserting a multiple partner half in the only global pin (real or virtual) of the same GRC. This special half will keep track of how many local pins it has to catch up. And each local pin will have its own local special half too. The reason for that is that local pins can join each other on several ways inside one GRC. Each time two local pins join, one of the two local halves will be kept on routing, and it will remind itself that it is carrying out a given number of local pins. This way, when this local half first reaches a special half of the unique global pin of the same net in this GRC, it can control how many local pins are still missing to connect.

This is the basic scheme. In practice, local pins of a given GRC will try to join any half of the same net, not necessarily the global pin of their same GRC. When it happens, the half they are supposed to connect to must be located and updated. Since we know the GRC in question, this is a trivial question of implementation.

## 4. Results and Conclusions

We have so far implemented the data structures, input and output, and LEGAL steps 1 and 2, which are sufficient to make simple routing of testing nets to check that the mechanism is working. The routing of small examples presents no problems, and demonstrates all the mechanisms described in this paper. Although we have not tested the implementation yet with real and large circuits, we observe that the mechanisms developed solve the problems they are supposed to.

The main contribution of this paper is to present the solutions for the questions stated above, which were being sought for a long time, since the first proposal of a LEGAL algorithm in 1994. At this point of view, we have achieved good results as a proof of concept. However, the implementation must be completed with other optimizations, which are not complicated, and tested in real examples. One step that is scheduled but still missing is to implement the needed algorithm to assign actual positions for virtual pins.

## 5. Acknowledgements

## 6. References

[1] Carro, Luigi et al. Ambiente ÀGATA de Projeto Versão Beta 2.0 In: Tercer Workshop Iberchip. Proceedings… México, 1997. p.495-503.

[2] P.E. Hart, N.J. Nilsson, and B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems, Science and Cybernetics, SSC-4 (1968), pp.100-107.

[3] Hashimoto, A.; Stevens, J. Wire Routing by Optimizing Channel Assignment within Large Apertures. DESIGN AUTOMATION WORKSHOP, 8., June 1971, Atlantic City. Proceedings... New York: ACM, 1971. p.155-163.

[4] Johann, Marcelo O. Roteamento sobre Células Transparentes. Porto Alegre, Pós-Graduação em Ciências de Computação/UFRGS, 1994. (Master Dissertation).

[5] Johann, M.; Reis, R. A Full Over-the-Cell Routing Model. IFIP VLSI'95, Chiba, Japan, 29 ago.-1 set., 1995. Proceedings...

[6] Johann, M.; Estrutura de Roteamento em Circuitos VLSI. Porto Alegre: CPGCC da UFRGS, 1997. EQ-15 (Exame de Qualificação). 80p.

[7] Johann, M.; Projeto Funcional do Roteador GAROTA. Porto Alegre: CPGCC da UFRGS, 1997. TI-649 (Trabalho Individual II). 54p.

[8] Johann, Marcelo, Carro, Luigi, Reis, Ricardo. Functional Design of GAROTA: Gate Array Router of ÀGATA System. SBCCI 1997. Proceedings… Gramado: UFRGS, 1997. p.21-30.

[9] Johann, Reis, R. Net by Net Routing with a New Path Search Algorithm. SBCCI 2000, Proceedings... Los Alamitos, IEEE, 2000.

[10] Johann, Marcelo. Novos Algoritmos para Roteamento de Circuitos VLSI. Porto Alegre: PPGC da UFRGS, 2001. (Doctoral Thesis)

[11] Johann, Marcelo; Reis, Ricardo. LEGAL: An Algorithm for Simultaneous Net Routing. SBCCI 2001, Proceedings… Los Alamitos: IEEE, 2001. p.180-185.

[12] Kao, Wen-Chung, Pargn, Tai-Ming. Cross Point Assignment With Global Rerouting for General Architecture Designs. IEEE Transactions on CAD of ICs and Systems. v.14, n.3, p.337. 1995.

[13] Lee, C. An Algorithm for Path Connections and its Applications. Transactions on Electronic Computers, New York, v.ec-10, p.346-365, Sept. 1961.

[14] Rivest, Ronald L.; Fiduccia, Charles M. A Greedy Channel Router. In: 19th DAC, June 1992, Las Vegas. Proceedings... New York: ACM/IEEE, 1982. p.418-424.

[15] Sherwani, Naveed; Algorithms for VLSI Physical Design Automation. Massachusetts: Kluwer, 1993. 538p.

[16] Sherwani, Naveed; BHINGARDE, Siddharth; PANYAM, Anand. Routing in the Third Dimension: From VLSI Chips to MCMs. IEEE, New York, 1995.