CHANNEL ROUTING WITH CONSTRAINT LOGIC PROGRAMMING AND DELAY

Neng-Fa Zhou

Faculty of Computer Science and System Engineering
Kyushu Institute of Technology
Iizuka, Fukuoka, Japan, 820.
E-mail: zhou@mse.kyutech.ac.jp

ABSTRACT

Channel routing is a well-known NP-complete problem in VLSI design. The problem is to find routing paths among a group of terminals that satisfy a given connection requirement without overlapping each other. This problem can be regarded as a constraint satisfaction problem. For a HV channel where there is only one horizontal layer and one vertical layer, the problem can be described easily in finite-domain constraint logic programming languages. However, for a nHV channel where $n > 1$, the modelization is not so straightforward because some entailment constraints are involved. We use delay to implement the entailment constraints. The resulting program is very simple, but demonstrates good performance that is comparable to that of previous programs.

1. INTRODUCTION

VLSI layout design consists of two phases: the first phase, called *placement*, determines the positions of the modules on the VLSI chip, and the second phase, called *routing*, connects the modules with wiring. *Channel routing* is a kind of routing where the routing area is restricted to a rectangular channel. A channel consists of two parallel rows with terminals on them. A connection requirement, called a *net*, specifies the terminals that must be interconnected through a routing path. A channel routing problem is to find routing paths for a given set of nets in a given channel such that no paths overlap each other [Bur86]. There are a lot of different definitions of the problem that impose different restrictions on the channel and routing paths. Figure 1 shows an example. There are two nets in the problem, N_1 requiring t_1 (the first terminal of the top row) and b_3 (the third terminal of the bottom row) be connected, and N_2 requiring b_1 and b_2 be connected. The routing path for each net consists of one horizontal segment placed on some track and several vertical segments. In a single-layer channel, only planer problems are routable. For example, if N_2 requires b_1 and t_2 be connected, then the problem is unsolvable. In practice, there are multiple layers available in a channel for routing. Figure 2 shows a solution for the example in a two-layer channel.

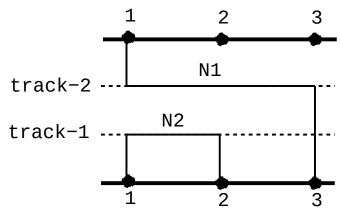

FIGURE 1 Single-layer channel.

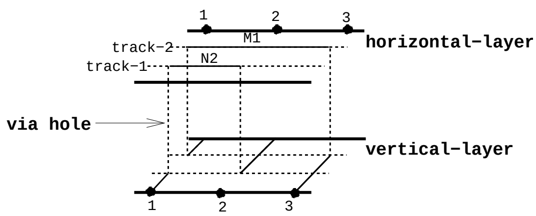

FIGURE 2 Two-layer channel.

The problem has been studied extensively in the VLSI design community since Hashimoto and Stevens [HS71] proposed it in 1971. Lapaugh has proved that the problem is NP complete [Lap80]. Many algorithms have been proposed for the problem [Deu76, FFL92, LSS94, Tak92, YK82]. Most of the traditional algorithms are based on graph algorithms. Recently, several modern heuristic search algorithms, for example, neural networks [Tak92], simulated annealing [BB88] and genetic algorithms [LSS94], have been proposed.

The channel routing problem is a finite-domain constraint satisfaction problem (CSP) [Mac86]. We concentrate on the *dogleg-free multi-layer* channel routing problem where the routing path for every net consists of only one horizontal line segment parallel to the two rows of the channel and several vertical line segments perpendicular to the two rows, and the routing area in the channel is divided into several pairs each of which consists of a horizontal layer for horizontal segments and a vertical layer for vertical segments. For each net, we need to determine the horizontal layer and the track on which the horizontal segment lays. The vertical layer for the net is determined automatically to be the one in the same pair as the horizontal layer. For this problem, each net to be routed can be treated as a variable whose domain is a set of all pairs of layers and tracks. To minimize the routing area means to to minimize the number of tracks.

Simonis [Sim90] has applied CHIP [DVS+88], a constraint logic programming language, to two-layer and three-layer channel routing problems where there is only one vertical layer involved. In [Zho95], we have implemented a forward checking algorithm that uses a special data structure called state tables. In this paper, we give a program for solving multilayer channel routing problems. The program uses the finite-domain constraint solving and delay facilities. It is very simple, but demonstrates good performance comparable to previous programs for the the Deutsch's difficult problem. Duchier and Huitouze have written a program [DH96] in CLP(FD) that is based on the same idea and demonstrates similar performance.

This paper is organized as follows: In Section 2, we define the channel routing problem in detail. In Section 3, we describe the program. In Section 4, we give the experimental results. In section 5, we compare our approach with other approaches and discuss the directions for improving the program.

2. CHANNEL ROUTING

A *channel* consists of two parallel horizontal rows with terminals on them. The terminals are numbered 1, 2, and so on from left to right. A *net* is a set of terminals that must be interconnected through a *routing path*. The *channel routing problem* is to find routing paths for a given set of nets in a given channel such that no segments overlap each other, and the routing area and the total length of routing paths are minimized.

$$
\begin{aligned}
N_1 &= \{t_2, t_5\}\\
N_2 &= \{b_1, b_6\}\\
N_3 &= \{b_2, b_4\}\\
N_4 &= \{t_3, t_9\}\\
N_5 &= \{b_3, t_4, b_5\}\\
N_6 &= \{t_6, b_7\}\\
N_7 &= \{t_7, b_{11}\}\\
N_8 &= \{b_8, b_{10}\}\\
N_9 &= \{b_9, t_{10}, b_{12}\}\\
N_{10} &= \{t_{11}, t_{12}\}
\end{aligned}
$$

FIGURE 3 The set of nets in the example problem.

There are a lot of different definitions of the problem that impose different restrictions on the channel and routing paths. We consider the *dogleg-free multi-layer* channel routing problem which impose the following three restrictions: First, the routing path for every net consists of only one horizontal segment that is parallel to the two rows of the channel, and several vertical segments that are perpendicular to the two rows. This type of routing paths is said to be *dogleg-free*. Second, the routing area in a channel is divided into several pairs of layers, one called a *horizontal layer* and the other called a *vertical layer*. Horizontal segments are placed in only horizontal layers and vertical segments are placed in only vertical layers. The ends of segments in a routing path are connected through *via holes*. There are several tracks in each horizontal layer. Minimizing the routing area means minimizing the number of tracks. Third, no routing path can stretch over more than one pair of layers. Thus, for each net, we only need to determine the horizontal layer and the track for the horizontal segment. The positions for the vertical segments are determined directly after the horizontal segment is fixed. In the following, we use nHV to denote a $2 \times n$-layer channel that has n pairs of horizontal and vertical layers.

For example, Figure 3 shows a set of nets. The terminals on the ith column of the top and bottom rows are denoted as t_i and b_i respectively. Figure 4 depicts a 2HV channel and the routing paths for the nets.

Two constraint graphs are created based on the given set of nets: one directed graph called a *vertical constraint graph* G_v and one indirected graph called a *horizontal constraint graph* G_h. In G_v, each vertex corresponds to a net and each arc from vertex u to vertex v means that net u must be placed *above* net v if they are placed in the same horizontal layer. The

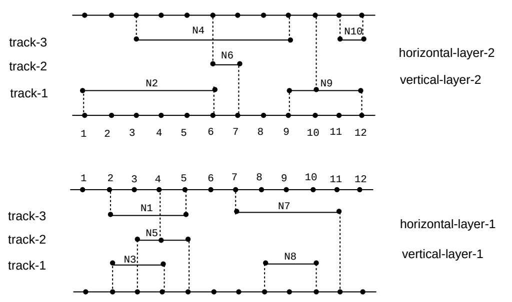

FIGURE 4 One solution for the example.

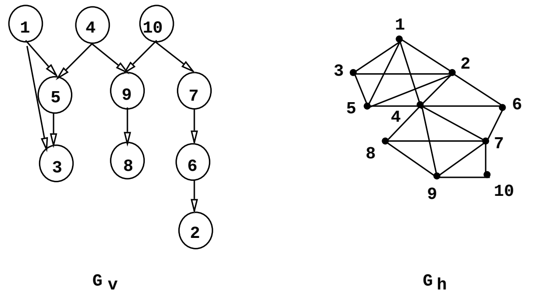

FIGURE 5 Constraint graphs.

relation above does not necessarily reflect the physical configuration of tracks. A track with number t_1th is said to be above another track with number t_2 in the same layer if t_1 is greater than t_2. In G_h, each vertex corresponds to a net and there is an edge between two vertices u and v if net u and net v cannot be placed on the same track. Figure 5 depicts the constraint graphs for the set of nets shown in Figure 3. There is an arc from vertex 1 to vertex 5 in G_v because t_5 is included in N_1 and b_5 is included in N_5. If N_5 is placed above N_1 or on the same track as N_1 in the same horizontal layer, then the vertical segments on the fifth column will overlap. There is an edge between vertex 1 and vertex 2 in G_h because the segment $(2, 5)$ connecting the two farthest terminals in N_1 and the segment $(1, 6)$ connecting the two farthest terminals in N_2 overlap each other. Notice that the relation *above* is not transitive unless there is only one vertical layer in the given channel. For example, in a 2HV channel, N_4 can even be placed below N_8 in the same horizontal layer if N_9 is placed in a different horizontal layer.

The *depth* of a net u in G_v is computed as follows: If u lies at the top of G_v, then $u's$ depth is 0; otherwise, suppose u has n predecessors v_1, v_2, …, v_n, then u's depth is $\max(\{d_{v1},\ldots,d_{vn}\})$ where d_v denotes the depth of v. There may exist cycles in G_v. In this case, all the vertices in a cycle have the same depth. The *length* of a routing path is the sum of the lengths of the horizontal and vertical segments in the path. For a horizontal segment whose left-most terminal number is l and whose right-most terminal number is r, the length of the segment is $r - l + 1$. Let t be the number of tracks in each horizontal layer. The length of a vertical segment between the ith track and the top row is $t - i + 1$ and the length of a vertical segment between the ith track and the bottom row is i.

3. PROGRAM

Channel routing problem is a CSP: Each net is treated as a variable whose domain is the set of all pairs of layers and tracks. The constraints are represented by the two constraint graphs G_v and G_h. Simonis [Sim90] has applied CHIP to the two-layer and three-layer channel routing problems where there is only one vertical layer involved. The relation *above* is represented as disequalities ($>$). However, as we have described in Section 2, the relation *above* is not transitive for general multi-layer channel routing problems and thus cannot be represented as disequalities. We present now a program in CLP(FD) for the problem that uses the couroutining facility.

The formulation described above does not directly suit CLP(FD) because the domains in CLP(FD) are restricted to sets of atomic values. To make the formulation suitable to CLP(FD), we concentrate on the tracks and number them uniquely as 0, 1, and so on. In this way, we can still associate each net with a domain variable that indicates the global track number. After the global track number is determined, the layer and the track in the layer can be easily computed. Let L be the number of horizontal layers and T be the number of tracks in each horizontal layer. The domains of all variables are $0..L \times T - 1$. Let t be the global track number for a net. The layer is $V//T + 1$ and the track number in the layer is $t - (t//T) \times T + 1$, where the operator // denotes integer division.

Figure 6 shows the program. The call `generate_vars(N,L,T,Vars)` generates `N` variables whose domains are $0..L \times T - 1$ and each of which corresponds to a net. The call `generate_constraints(Vars,T)` generates constraints among the domain variables. The call `label(Vars)` assigns values to variables.

It is very straightforward to generate horizontal constraints. For each pair of variables `X` and `Y`, if they cannot lie on the same track, then emit the inequality constraint $X \neq Y$.

```
route(N,L,T):-
      generate_vars(N,L,T,Vars),
      generate_constraints(Vars,T),
      label(Vars).

label([]):-!.
label(Vars):-
      choose(Vars,Var,Rest),
      indomain(Var),
      label(Rest).
```

FIGURE 6 Program in CLP(FD).

```
different_track(X,Y):-X≠Y.
```

However, generating vertical constraints is not so straightforward. Suppose there is an arc between two variables X and Y in Gv. The following entailment constraint declaratively specifies the above relation: "if X and Y lie in the same layer, then X must be greater than Y". Procedurally, the constraint can be described as follows:

```
delay above(X,Y,T) if var(X), var(Y).
above(X,Y,T):-
          nonvar(X),!
          Y ∉ X..(X//T)*T+T-1.
above(X,Y,T):-
          X ∉ (Y//T)*T..Y.
```

Figure 7 illustrates the relation. If both X and Y are variables, then delay the constraint; if X has gotten a value, then Y cannot be routed in the shadow area $X..(X//T)*T-1$; if Y has gotten a value, then X cannot be routed n the shadow area $(Y//T)*T..Y$.

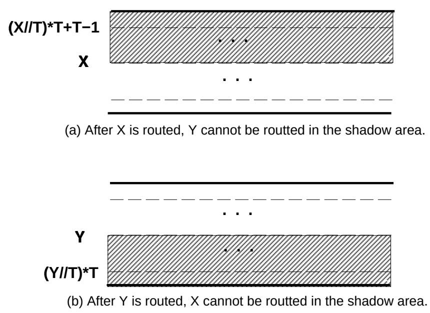

FIGURE 7 The constraint *X is above Y*

.

4. EXPERIMENTAL RESULTS

In this subsection, we present the results for the Deutch's problem obtained with B-Prolog and compare them with the best results known now.

4.1 B-PROLOG

B-Prolog is an emulator-based Prolog system. Its performance is comparable in general to that of emulated SICStus-Prolog (version 2.1), a commercial system developed at Swidish Institute of Computer Science. For a group of search programs that do a lot of backtracking, B-Prolog is about forty percent faster than emulated SICStus-Prolog.

The finite-domain constraint solver is mostly written in canonical-form Prolog where input and output unifications are separated and determinisms of clauses are denoted explicitly. The performance of the constraint solver is better than that of clp(FD) (version 2.2), a system developed at INRIA, and Eclipse (version 3.5.1), a system developed at ECRC. For the 64-queens problem, B-Prolog takes 0.8 second on a SPARC-10, whereas Eclipse takes 1.6 seconds and clp(FD) takes 2.8 seconds on the same computer.

4.2 PROGRAMS AND BENCHMARKS

The program can minimize the number of tracks and the total length of routing paths by using branch & bound. It is only around 350 lines long excluding comments, blanks, the data for the nets, and the code for displaying solutions.

The Deutsch's difficult problem is used as the benchmark. The benchmark suite given in [YK82] are well used in the VLSI design community, among which the Deutsch's difficult problem is a representative one. The problem is to route a set of 72 nets on a channel where there are 174 terminals on each row. There are 117 arcs in the constraint graph G_v and 846 edges in G_h.

4.3 HEURISTICS

The order in which variables are instantiated can affect the efficiency of the algorithm dramatically. We use the following rules to choose a variable.

1. Choose first a variable whose corresponding net lies at the bottom in G_v.

2. Choose first a variable with the smallest domain.

3. Choose first a variable whose corresponding net has the greatest degree in G_v.

	HV	2HV	3HV	4HV
Initial bound	40	20	14	10
Best solution	28	11	7	5

Table 1 The best solutions found in five minutes.

4. Choose first a variable whose corresponding net has the greatest degree in G_h.
5. Choose first a variable whose corresponding net lies at the bottom of G_v.

The first rule ensures that the nets at the bottom of G_v are routed before those above them. All the other rules are consistent with the *first fail principle* [Hen889]. Choosing first a variable that has the smallest domain and participates in the largest number of constraints can usually make a failure occur earlier. The second rule is only used in the forward checking algorithm.

4.4 SOLUTIONS

We have run the programs on a SPARC-10 many times by asking them different questions.

Question 1:

What solutions for HV, 2HV, 3HV and 4HV channels can be found in five minutes that require the minimum numbers of tracks.?

Table 1 shows the answers. The row *Initial bound* depicts the initial bound on the number of tacks, and the row *Best solution* gives the best solution obtained in five minutes.

We have tried several combinations of rules for choosing variables. The combination 1-5-3-4 demonstrates the best performance for all the programs and all the types of channels. The best HV solution obtained is known to be optimal in terms of the number of tracks [KSP73]. The optimal solutions for the remaining types of channels have not yet been reported.

The CHIP program described in [Sim90] found an optimal HV solution for the same problem in less than 30 seconds. Takefuji's programs found the same best solutions. The router described in [FFL92] found in less than one second a solution for 2HV that requires 10 tracks, but it does not require segments in a routing path to be in only one pair of layers.

Question 2:

Are the best solutions optimal? The program failed to prove the optimality of the solutions in 12 hours.

HV	2HV	3HV	4HV
1.9	1.8	2.0	26.1

Table 2 The times (in seconds) required to find the first best solutions.

HV	2HV	3HV	4HV
5954	4102	3567	3364

Table 3 The lengths of the best solutions found in one hour.

Question 3:

How many seconds does it take to find the first best solution for each type of channel? Table 2 shows the answers. The heuristics 1-5-3-4 is used.

Question 4:

What shortest solutions can be found in one hour? Table 3 shows the lengths of the solutions. Figures ?? to ?? show the solutions for HV and 4HV channels.

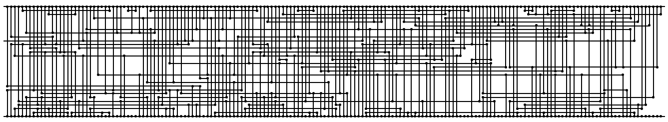

FIGURE 8 Number of tracks = 28, Length=5954

5. CONCLUSION

This this paper, we described a program for the channel routing problem in a finite-domain constraint programming language. There have been a huge number of algorithms proposed to solve the channel routing problem. Recent algorithms tend to be very complicated and thus are very difficult to implement. Furthermore, when the restrictions on the channel or routing paths change, the algorithms must be redesigned. Compared with these traditional algorithmic approaches, our approach is declarative and very simple. The program can be easily adapted to other types of routing problems by modifying the definitions of domains, constraints and heuristics.

The program can be improved in several directions. Firstly, the current program only use general heuristics for ordering variables. It would be more efficient

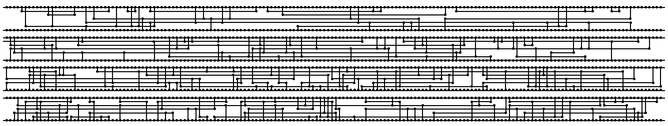

FIGURE 9 Number of tracks = 5, Length = 3364.

to use some problem specific heuristics used in traditional algorithms. For example, such information about nets concerning the lengths of nets, types of nets (two-terminal nets, multi-terminal nets, nets connecting only terminals at the top, nets connecting only terminals at the bottom, etc.) can be used to choose variables. Secondly, the program can be improved by introducing heuristics for choosing appropriate values for selected variables. These two improvements should be justified by experiments. For this purpose, a large number of benchmark problems must be tested. Thirdly, the program can be executed in parallel on a multi-processor computer or a network of computers. Parallel search is a promising technique that can be used to find good solutions and prove optimality of solutions.

ACKNOWLEDGEMENTS

The idea originated from my early work described in [Zho95] and discussion with Dr. S.L. Huitoze and Dr. D. Dechier who contributed the key idea of using delay.

REFERENCES

[BB88] Brouwer, R.J. and Banerjee, P., A Parallel Simulated Annealing Algorithm for Channel Routing on a Hypercube Multiprocessor, in: *IEEE Int. Conf. Comput. Design*, 1988, 4-7.

[Bur86] Burstein, M, Channel Routing in:*Layout Design and Verification*, North-Holland, 1986, 133-167.

[DH96] Duchier, D. and Huitouze, S.L.: Channel Routing with CLP(FD), submitted to PACT'96, 1996.

[Deu76] Deutsch, D.N., A Dogleg Channel Router, in: *Proc. 13th Design Automation Conference*, 1976, 425-433.

[DVS+88] Dincbas, M., Van Hentenryck, H., Simonis, H., Aggoun, A., Graf, T., and Berthier, F., The Constraint Logic Programming Language CHIP, in: *Proc. FGCS'88*, 1988, 693-702.

[FFL92] Fang, S.C., Feng, W.S., and Lee, S.L., A New Efficient Approach to Multilayer Channel Routing Problem, in: *Proc. of the 29th ACM/IEEE Design Automation Conference*, 1992, 579-584.

[HS71] Hashimoto, A and Stevens, S., Wire Routing by Optimizing Channel Assignment within Large Apertures, in: *Proc. 8th Design Automation Workshop*, 1971, 155-169.

[KSP73] Kernighan, B.W., Schweikert, D.G., and Persky, G., An Optimum Channel-routing Algorithm for Polycell Layouts of Integrated Circuits, in: *Proc. 10th Design Automation Workship*, 1973, 50-59.

[Kum92] Kumar, V., Algorithms for Constraint Satisfaction Problems: A Survey, in: *AI Magazine*, 1992,32-44.

[Lap80] LaPaugh, A.S., *Algorithms for Integrated Circuits Layout: An Analytic Approach*, PhD Dissertation, MIT Lab. of Computer Science, 1980.

[LSS94] Liu, X., Sakamoto, A. and Shimamoto, T., Genetic Channel Router, in: *IEICE Trans. Fundamentals*, 1994, E77-A:492-501.

[Mac86] Mackworth, A., Constraint Satisfaction, in: *Encyclopedia of Artificial Intelligence*, John Wiley & Sons, 1986, 205-211.

[Sim90] Simonis, H., *Channel Routing Seen as a Constraint Problem, Tech. Rep.*, TR-LP-51, ECRC, Munich, July, 1990.

[Tak92] Takefuji, , *Neural Network Parallel Computing*, Kluwer Academic Publishers, 1992.

[Hen889] Van Hentenryck, P., *Constraint Satisfaction in Logic Programming*, MIT Press, 1989.

[YK82] Yoshimura, T. and Kuh, E.S., Efficient Algorithms for Channel Routing, in: *IEEE Trans. CAD*, 1:25-35 (1982).

[Zho95] Zhou, N.F., A Logic Programming Approach to Channel Routing, in: *Proc. 12th International Conference on Logic Programming*, MIT Press, 159-173, 1995.