Constraint-Based Dogleg Channel Routing with Via Minimization

I-Lun Tseng[1], **Huan-Wen Chen**[1], **Che-I Lee**[1], **and Adam Postula**[2]
[1]Department of Computer Science & Engineering, Yuan Ze University, Taiwan, R.O.C.
[2]School of Information Technology & Electrical Engineering, The University of Queensland, Australia

Abstract - *In this article, we present an algorithm which is capable of transforming a gridded dogleg channel routing problem into a constraint programming (CP) problem. The transformed CP problem consists of a set of variables and a list of constraints; it can be solved by JaCoP, a finite-domain constraint programming solver. For a given dogleg channel routing problem, our approach is able to minimize the number of tracks and the number of vias. Although the transformed CP problems cannot be solved in polynomial time, optimal results can be found efficiently for small to medium cases. Moreover, for large cases, suboptimal results can be generated in exchange for significantly reduced execution time. As constraint programming technologies advance (e.g., parallel constraint programming), the execution time of the proposed approach can be improved. Additionally, our approach has the potential to be used in dealing with other routing problems in VLSI physical design automation.*

Keywords: Dogleg Channel Routing, VLSI Physical Design Automation, Constraint Programming

1 Introduction

Channel routing is a type of problems arising in the detailed routing phase of VLSI physical design automation [1] as well as in the design of printed circuit boards (PCBs) [2]. Although channel routing has not been an active research field in recent years, the use of constraint programming technologies in solving this type of problems has not been completely investigated. Moreover, gaining full understanding of these fundamental problems is essential to the research and development of other routing algorithms [3].

In order to solve a channel routing problem, many routing algorithms generate horizontal and/or vertical constraints for the problem [4, 5]. For a channel routing problem containing cyclic vertical constraints, doglegs are required in order to complete the routing [5]. Since dogleg channel routing problems are NP-complete [6], many heuristic algorithms have been developed and proposed [5, 7, 8]. Unfortunately, those heuristic algorithms are not guaranteed to generate optimal solutions.

Instead of developing heuristic routing algorithms, we transform a gridded dogleg channel routing problem into a constraint programming (CP) problem. The transformed CP problem can then be solved optimally by a constraint programming solver. As a result, the number of tracks can be minimized. With this minimum number of tracks, furthermore, the number of vias can also be minimized.

Constraint programming [9] is a type of declarative programming paradigm in that it allows users to specify a problem in terms of variables and constraints over those variables; a constraint programming solver can then be used to find the solution(s) to the specified problem. JaCoP (Java Constraint Programming [10]) is an open-source constraint programming library which was implemented in Java. The JaCoP library contains a number of API functions as well as a built-in constraint programming solver. After users specify a problem via those API functions, the solver can find the solution(s) to the problem (if the problem has at least one solution). JaCoP has been used in solving many difficult problems (such as optimization problems, scheduling and resource assignment problems [11, 12], and problems of partitioning parameterized polygons [13]), although the time complexity for solving those problems may not be polynomial.

A channel routing problem can be considered as a multi-objective optimization problem, as we may need to simultaneously optimize two or more objectives, such as minimizing the number of tracks [7], minimizing the number of vias [14], minimizing the crosstalk [15, 16], and minimizing the total wire length [17]. Most of heuristic routing algorithms only consider one or two of those objectives, and adding other objectives may result in redesign of those algorithms. Although this article focuses on the objectives of minimizing the number of tracks and the number of vias, our approach can be further extended to consider other objectives (e.g., crosstalk minimization).

The use of constraint programming technologies in solving channel routing problems is not new. In [18], the integration of constraint programming and evolution programs has been used to solve dogleg-free multilayer channel routing problems. In [19], Phillips proposed the adoption of constraint logic programming in solving dogleg channel routing problems. Our approach differs from the one presented in [19] in that our approach requires different (usually simpler) types of constraints. In addition, our approach is capable of minimizing the number of vias.

The rest of this paper is organized as follows. In Section 2, we formulate dogleg channel routing problems that we

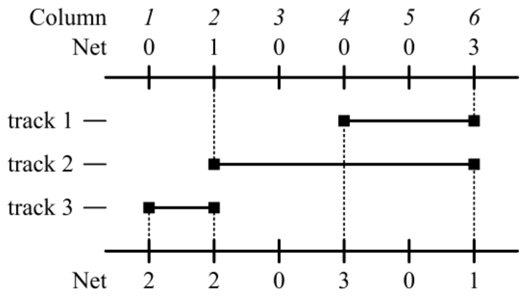

Figure 1. A (gridded) channel routing problem and one of its solutions without using doglegs

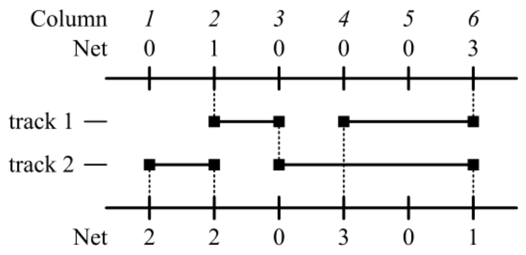

Figure 2. A solution to the channel routing problem (shown in Figure 1) with the use of doglegs

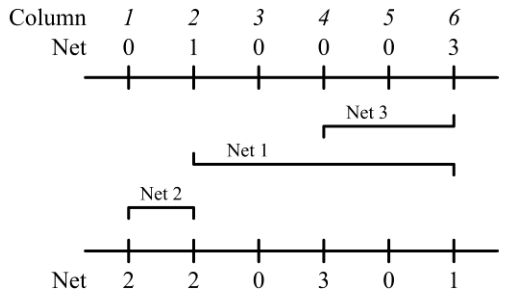

Figure 3. Horizontal span of each net

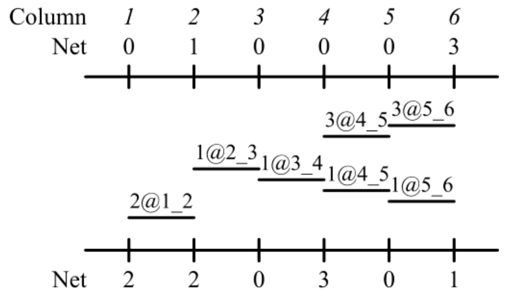

Figure 4. Horizontal wire fragments and their representations

intend to solve. Horizontal and vertical constraints are discussed in Sections 3 and 4, respectively. An optional function which is capable of minimizing the number of vias is presented in Section 5. Our algorithm for solving gridded dogleg channel routing problems is described in Section 6. Experimental results are presented in Section 7. Finally, conclusions are drawn in Section 8.

2 Problem Formulation

In a channel routing problem, a channel is a rectangular region bounded by two parallel rows (the top row and the bottom row). The two parallel rows have terminals and each terminal has a number, which represents the name of a net. Terminals having the same number must be connected together, except that terminals with the number zero require no connection.

In this paper, it is assumed that a channel routing problem has only two routing layers, one layer for horizontal wire segments and the other for vertical wire segments. Endpoints of wire segments must be located within the channel (the rectangular region). For the wire segments that reside on different layers, in addition, they can be connected by *vias*. In the figures in this paper, vias are denoted by small black squares.

Figure 1 shows an example of a channel routing problem and one of its solutions. The problem has six columns and each terminal lies at the intersection of a row and a column. Moreover, horizontal wire segments, which are used for routing purposes, must lie on the tracks. As can be seen in this example, the routing solution uses three tracks. The columns, rows, and tracks form an array of (virtual) grids. Therefore, the channel routing problem shown in Figure 1 is a *gridded* channel routing problem if all the endpoints of (horizontal and vertical) wire segments are restricted to lie on the grids.

In a channel routing problem, since the width of the channel is fixed, minimizing the routing area is equivalent to minimizing the number of tracks (the height of the channel). By introducing doglegs [20] in solving the problem shown in Figure 1, it is possible to complete the routing with only two tracks, as shown in Figure 2. The use of doglegs in solving channel routing problems is a technique of great importance. In the cases where cyclic vertical constraints exist [5], doglegs must be used in order to complete the routing.

In our model of a gridded dogleg channel routing problem, each net is composed of a number of horizontal and vertical wire segments. In addition, these horizontal wire segments must be placed between the net's leftmost column and rightmost column. For the channel routing problem given in Figure 1, the horizontal span of each net is shown in Figure 3. Based on the horizontal spans, a number of horizontal wire *fragments* (or smaller horizontal wire segments), as shown in Figure 4, can be generated by cutting the horizontal spans into pieces. Each of these horizontal wire fragments spans between two adjacent columns. In addition, the union of all the horizontal wire fragments of one net must cover the total horizontal span of the net. The name of each horizontal wire fragment is coded as follows (as the example shown in Figure 4):

<net name>@*<left column no.>*_*<right column no.>*

In our model of a channel routing problem, each horizontal wire fragment is associated with a numerical value; the value represents the track on which the wire fragment is

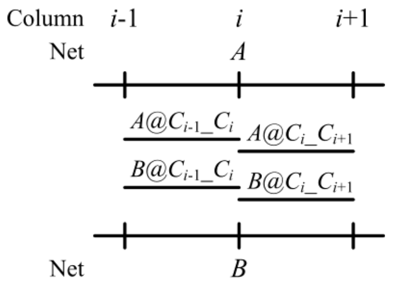

Figure 5. An example illustrating case 1 in the process of generating vertical constraints

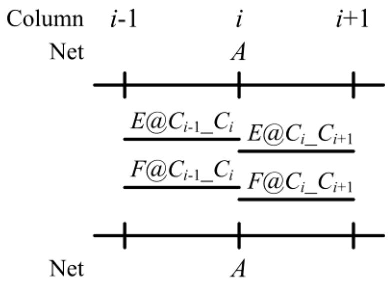

Figure 6. An example illustrating case 2 in the process of generating vertical constraints

located. In our algorithm, moreover, vertical wire segments are not cut into fragments; positions of vertical wire segments can be decided easily after all the horizontal wire fragments have been placed. For instance, the routing result shown in Fig . 2 can be represented by the following code:

[1@2_3=1, 1@3_4=2, 1@4_5=2, 1@5_6=2, 2@1_2=2, 3@4_5=1, 3@5_6=1]

3 Horizontal Constraints

Since two horizontal wire fragments belonging to different nets cannot overlap, these two fragments must be assigned different track numbers if they belong to the same column interval. We can thus use the following inequality to specify this type of constraints in a channel routing problem:

$$net_A@C_i\_C_{i+1} \neq net_B@C_i\_C_{i+1}$$

where net_A and net_B are the names of two different nets, and columns C_i and C_{i+1} denote two adjacent columns. For the example shown in Figure 4, therefore, the following horizontal constraints must be generated:

$$3@4\_5 \neq 1@4\_5$$
$$3@5\_6 \neq 1@5\_6$$

If there are many different horizontal wire fragments between columns C_i and C_{i+1}, generating unequal constraints for all pairs of these fragments might be cumbersome. Therefore, the JaCoP function `Alldifferent(`*list-of-nets*`)` is used in order to reduce the number of constraints.

4 Vertical Constraints

In our algorithm, the process of generating vertical constraints for a channel routing problem involves looping through all the columns from left to right. Also, vertical constraints are generated according to different cases at each column.

We define a number of terms before detailing each case of generating vertical constraints. When a column (i) is encountered, the terminal at the intersection of the top (bottom) row and column i is called the *upper* (*lower*) *terminal* of column i. Furthermore, if the upper (lower) terminal belongs to net A, then net A can be referred to as the *upper* (*lower*) *net* of column i.

4.1 Case 1

Case 1 of generating vertical constraints arises at a column where each of the top and bottom row contains a terminal, and the two terminals belong to different nets. As the example shown in Figure 5, when column i is encountered, horizontal fragments of net A (which are touching column i) must be located higher than horizontal fragments of net B (which are also touching column i). Otherwise, the two nets (A and B) will overlap at the vertical column and result in a short circuit. In other words, the following constraints must be generated and satisfied:

$$A@C_{i\text{-}1}\_C_{\mathrm{i}} < B@C_{i\text{-}1}\_C_i$$
$$A@C_{i\text{-}1}\_C_{\mathrm{i}} < B@C_i\_C_{i+1}$$
$$A@C_i\_C_{i+1} < B@C_{i\text{-}1}\_C_i$$
$$A@C_i\_C_{i+1} < B@C_i\_C_{i+1}$$

At column i, however, if there is only one horizontal wire fragment belonging to the upper net or only one horizontal wire fragment belong to the lower net, only the constraints that contain valid horizontal wire fragments need to be generated.

For the example shown in Figure 4, when column 2 is encountered, the vertical constraint "1@2_3 < 2@1_2" will be generated. Also, the constraint "3@5_6 < 1@5_6" will be generated when column 6 is encountered.

4.2 Case 2

In the process of generating vertical constraints, case 2 occurs when the same upper and lower net can be found at a column (i). As the example shown in Figure 6, in order to connect terminals of net A, there will be a vertical wire segment which directly connects the upper and lower terminals of column i. As a result, all other nets (nets E and F in this example) passing through column i must not have doglegs at the column. In this example, therefore, the following constraints need to be generated:

$$E@C_{i\text{-}1}\_C_{\mathrm{i}} = E@C_i\_C_{i+1}$$
$$F@C_{i\text{-}1}\_C_{\mathrm{i}} = F@C_i\_C_{i+1}$$

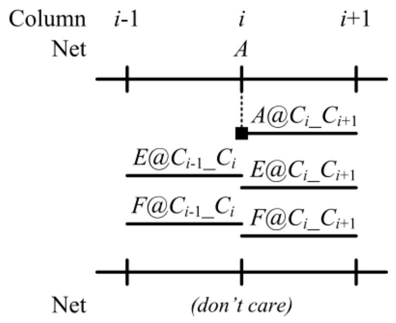

Figure 7. An example illustrating case 3 in the process of generating vertical constraints

Please note that case 1 and case 2 of generating vertical constraints are mutually exclusive.

4.3 Case 3

Case 3 focuses on generating constraints for the situation when the upper or lower net of a column (i) interacts with all of other nets passing through the column. Figure 7 illustrates an example of this case. At column i, net A is the upper net and it is assumed that the horizontal fragment "$A@C_i\_C_{i+1}$" exists. In addition, nets E and F are assumed to be the nets passing through column i. Please note that we do not need to consider whether the lower net exists or not at the moment. As a result, the following two lines of constraints should be generated for the example shown in Figure 7:

- (($A@C_i\_C_{i+1} < E@C_{i-1}\_C_i$) AND ($A@C_i\_C_{i+1} < E@C_i\_C_{i+1}$)) OR ($E@C_{i-1}\_C_i = E@C_i\_C_{i+1}$)
- (($A@C_i\_C_{i+1} < F@C_{i-1}\_C_i$) AND ($A@C_i\_C_{i+1} < F@C_i\_C_{i+1}$)) OR ($F@C_{i-1}\_C_i = F@C_i\_C_{i+1}$)

In the above constraints, net E (or F) is allowed to have a dogleg at column i only when its two horizontal wire fragments are located lower than the horizontal wire fragment of net A; otherwise, the dogleg will overlap with the vertical wire segment of net A.

In Figure 7, if net A had both the horizontal fragments to the left and to the right of column i, there would be more lines of constraints. Also, more lines of constraints would be required if there were more nets passing through the column. In this case (case 3) of generating vertical constraints, please note that similar forms of constraints must be generated if the lower net of column i exists.

4.4 Case 4

In Figure 6 or 7, net E or F can have a dogleg at column i; it is also possible that both of the nets have doglegs at the column. Case 4 of generating vertical constraints considers these conditions in order to make sure that doglegs do not overlap at each column. In other words, for each pair of nets passing through a column (and those nets are not the upper or lower nets), their doglegs cannot overlap at the column. For the example shown in Figure 6 or 7, therefore, the following constraints must be generated.

- IF ($E@C_{i-1}\_C_i < E@C_i\_C_{i+1}$) THEN ($F@C_{i-1}\_C_i = F@C_i\_C_{i+1}$) OR ($F@C_{i-1}\_C_i < E@C_{i-1}\_C_i$ AND $F@C_i\_C_{i+1} < E@C_{i-1}\_C_i$) OR ($E@C_i\_C_{i+1} < F@C_{i-1}\_C_i$ AND $E@C_i\_C_{i+1} < F@C_i\_C_{i+1}$)
- IF ($E@C_i\_C_{i+1} < E@C_{i-1}\_C_i$) THEN ($F@C_{i-1}\_C_i = F@C_i\_C_{i+1}$) OR ($F@C_{i-1}\_C_i < E@C_i\_C_{i+1}$ AND $F@C_i\_C_{i+1} < E@C_i\_C_{i+1}$) OR ($E@C_{i-1}\_C_i < F@C_{i-1}\_C_i$ AND $E@C_{i-1}\_C_i < F@C_i\_C_{i+1}$)
- IF ($F@C_{i-1}\_C_i < F@C_i\_C_{i+1}$) THEN ($E@C_{i-1}\_C_i = E@C_i\_C_{i+1}$) OR ($E@C_{i-1}\_C_i < F@C_{i-1}\_C_i$ AND $E@C_i\_C_{i+1} < F@C_{i-1}\_C_i$) OR ($F@C_i\_C_{i+1} < E@C_{i-1}\_C_i$ AND $F@C_i\_C_{i+1} < E@C_i\_C_{i+1}$)
- IF ($F@C_i\_C_{i+1} < F@C_{i-1}\_C_i$) THEN ($E@C_{i-1}\_C_i = E@C_i\_C_{i+1}$) OR ($E@C_{i-1}\_C_i < F@C_i\_C_{i+1}$ AND $E@C_i\_C_{i+1} < F@C_i\_C_{i+1}$) OR ($F@C_{i-1}\_C_i < E@C_{i-1}\_C_i$ AND $F@C_{i-1}\_C_i < E@C_i\_C_{i+1}$)

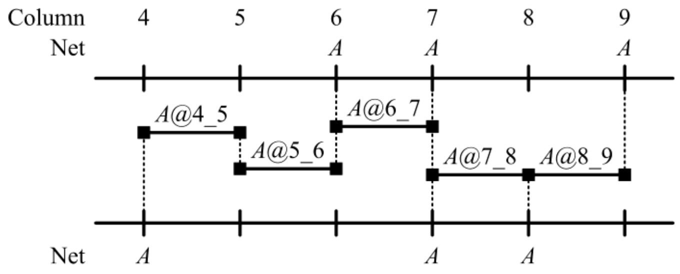

Figure 8. An example illustrating how to calculate the number of vias for a net

5 Minimizing the Number of Vias

In VLSI physical design, minimizing the number of vias can improve circuit performance and yield. Also, since our approach allows the use of doglegs in solving gridded channel routing problems, a number of extra vias may be induced in final routing results. It is thus desirable to reduce the number of vias in our approach. We have implemented the via minimization function by using constraint programming; it is an optional function and users can turn it on or off. The via minimization function is capable of minimizing the number of vias without increasing the number of tracks.

To calculate the number of vias for each net, additional variables are used. For a net whose leftmost column is l and rightmost column is r, we use a variable for each column between l and r to denote the number of vias at the column. The following type of variable names is used to represent the number of vias at a column:

<net name>@*<column no.>*

For the example shown in Figure 8, net A's leftmost column is 4 and its rightmost column is 9. Variable names

$A@4$, $A@5$, …, and $A@9$ are used to represent the number of vias at the corresponding column. Therefore, the total number of vias for net A is:

$$A@V = A@4 + A@5 + A@6 + A@7 + A@8 + A@9$$

where the variable name "$A@V$" denotes the total number of vias for net A. Also, it is trivial that $A@4 = 1$ and $A@9 = 1$ since columns 4 and 9 are the endpoints of net A.

In Figure 8, the following constraint will be generated by our algorithm when column 5 is encountered:

- IF ($A@4\_5 \neq A@5\_6$) THEN ($A@5 = 2$) ELSE ($A@5 = 0$)

That is because net A will have two vias at column 5 if there is a dogleg at the column. On the contrary, net A will not have any via at column 5 if there is no dogleg at the column. With this concept, we can generate the following constraints when column i is encountered, where i = 6, 7, or 8:

- IF ($A@C_{i-1}\_C_i \neq A@C_i\_C_{i+1}$) THEN ($A@C_i = 2$) ELSE ($A@C_i = 1$)

Finally, our algorithm generates the constraint below in order to calculate the total number of vias for a channel routing problem:

- Total_Num_of_Vias = $net1@V + net2@V + \dots$

where "Total_Num_of_Vias" is a variable. JaCoP provides an API function which is capable of minimizing the value of the variable while other constraints are satisfied.

6 The Algorithm

In our algorithm, a gridded dogleg channel routing problem is transformed into a constraint programming (CP) problem and then solved by JaCoP's solver. However, due to the fact that positions of horizontal wire fragments are treated as variables, the number of available tracks must be determined before the transformed CP problem can be solved. Since *channel density* is the minimum number of tracks required in order to solve a two-layer channel routing problem [8], our algorithm uses it as the initial value for the number of available tracks. Therefore, the domain for the position of each horizontal wire fragment is set to [1, *channel_density*]. The channel routing algorithm is detailed in Figure 9.

```
Algorithm CONSTRAINTBASEDCHANNELROUTER(TR, BR, VM)
Input.   The description to a gridded dogleg channel routing
         problem, which includes (1) TR, which is the list of
         terminals at the top row, (2) BR, which is the list of
         terminals at the bottom row. Also, a Boolean variable VM
         is used in order to control the activation of the via
         minimization function.
Output.  A permutation of horizontal wire fragments, from which a
         solution to the input channel routing problem can be
         constructed.
1.    D ← the channel density of the input channel routing
      problem
2.    Max ← D
3.    Do {
4.        Generate variables for horizontal wire fragments; the
          domain of each variable is set to [1, Max].
5.        Generate horizontal constraints for each column interval
          (Section 3).
6.        Generate vertical constraints for each column (Section 4).
7.        if VM equals TRUE then
8.            Generate variables and constraints for minimizing the
              number of vias (Section 5).
9.        Specify generated variables and constraints via JaCoP's
          API functions.
10.       Invoke JaCoP's solver to solve the specified constraint
          programming (CP) problem. When a solution has been
          found, report the solution and then exit the algorithm.
11.       Max ← Max + 1
12.   } while (JaCoP has not found a solution)
```

Figure 9. The Gridded Dogleg Channel Routing Algorithm

In the above algorithm, if the transformed CP problem cannot be solved by using the specified number of available tracks, the algorithm will increase the number of available tracks by 1 and then solve the transformed problem again. In our implementation of the algorithm, the value of *Max* can also be set manually. The algorithm stops when a solution has been found. Please note that all types of constraints (including IF-THEN and IF-THEN-ELSE constraints) mentioned in Sections 3, 4, and 5 can be specified by using JaCoP's API functions.

7 Experimental Results

A number of testcases have been used to test the correctness and to measure the performance of our program; some of the experimental results are shown in Table 1. Note that the table shows the results of our program with the via minimization function turned on. Testcases and routing results of "Figure 2" and "Figure 11" can be seen from Figure 2 and Figure 11, respectively. Testcases "Deutsch-1" and "Deutsch-2" were modified from the Deutsch's difficult example [3, 20]. All of the testcases were run on a PC with an Intel Q9550 CPU and 8 GB of RAM. For the routing result shown in Figure 2, our program generates the following code:

```
[1@2_3=1, 1@3_4=2, 1@4_5=2, 1@5_6=2, 2@1_2=2,
3@4_5=1, 3@5_6=1, 1@2=1, 1@3=2, 1@4=0, 1@5=0,
1@6=1, 1@V=4, 2@1=1, 2@2=1, 2@V=2, 3@4=1,
3@5=0, 3@6=1, 3@V=2, Total_Num_of_Vias=8]
```

Figures 10 and 11 illustrate the same routing problem with different routing results; one is with the via minimization function turned off and the other turned on.

Although our approach is capable of finding optimal solutions, the execution time can be very long. That is because dogleg channel routing problems are NP-complete. However, as can be seen in the Deutsch-2 case in Table 1, suboptimal results can be generated in exchange for significantly reduced execution time.

Table 1. Experimental Results of Constraint-Based Dogleg Channel Routing with Via Minimization

Testcase	# nets	# columns	C. Density	min. # tracks	# tracks	Tran. Time	Tot. Time	# vars	# c. lines
Figure 2	3	6	2	2	2	0.09 sec.	0.14 sec.	20	17
Figure 11	5	9	5	6	6	0.14 sec.	0.34 sec.	64	175
Deutsch-1	27	57	16	16	16	0.25 sec.	11.01 sec.	1,221	11,891
Deutsch-2	52	156	19	19	25	0.53 sec.	4.18 sec.	3,985	47,871
					24	0.53 sec.	5.10 sec.		
					23	0.53 sec.	> 40 hrs.		

# nets: the number of nets in the testcase; # columns: the number of columns in the testcase; C. Density: the channel density of the testcase; min. # tracks: the minimum number of tracks required for solving the channel routing (CR) problem; # tracks: the number of tracks used by our program for solving the problem (the value was assigned by setting the *Max* manually); Tran. Time: the CPU time for transforming the input CR problem into a CP problem; Tot. Time: the total CPU time for solving the CR problem (including Tran. Time); # vars: the number of variables in the transformed CP problem; # c. lines: the number of constraint lines in the transformed CP problem.

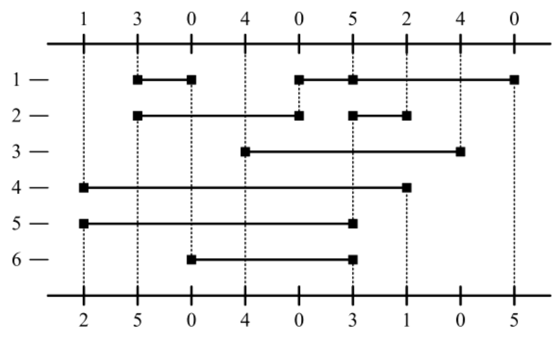

Figure 10. A minimum-track solution to a dogleg channel routing problem without via minimization (total number of vias = 17)

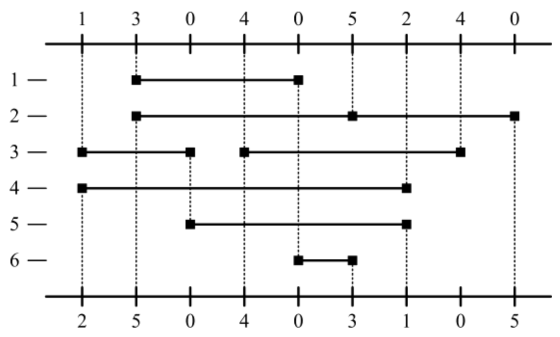

Figure 11. A minimum-track solution to the dogleg channel routing problem (shown in Figure 10) with the via minimization function turned on (total number of vias = 15)

8 Conclusion

We proposed an algorithm which is capable of transforming a gridded dogleg channel routing problem with via minimization into a constraint programming problem, and the transformed problem can be solved by a constraint programming solver. Our approach can be further extended to consider crosstalk and total wire length. Although the experimental results show that the running time of our approach cannot compete with many existing channel routers, optimal results can be generated for small to medium cases. In addition, for large cases, suboptimal results can be generated in exchange for significantly reduced running time. We believe that the performance of our approach can be improved with the advance of constraint programming technologies (eg, parallel constraint programming) and multi-core processors.

References

[1] Naveed A. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer Academic Publishers, 1999.

[2] Akihiro Hashimoto and James Stevens, “Wire Routing by Optimizing Channel Assignment within Large Apertures,” In *Proceedings of Design Automation Conference*, pp. 155-169, 1971.

[3] Rajat K. Pal, *Multi-Layer Channel Routing: Complexity and Algorithms*, Narosa Publishing House, 2000.

[4] Jia-Shung Wang and R. C. T. Lee, “An Efficient Channel Routing Algorithm to Yield an Optimal Solution,” *IEEE Transactions on Computers*, Vol. 39, No. 7, pp. 957-962, Jul. 1990.

[5] Takeshi Yoshimura and Ernest S. Kuh, “Efficient Algorithms for Channel Routing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. CAD-1, No. 1, pp. 25-35, Jan. 1982.

[6] Thomas G. Szymanski, “Dogleg Channel Routing is NP-Complete,” *IEEE Transactions on Computer-Aided Design*, Vol. CAD-4, No. 1, pp. 31-41, Jan. 1985.

[7] James Reed, Alberto Sangiovanni-Vincentelli, and Mauro Santomauro, “A New Symbolic Channel Router: YACR2,” *IEEE Transactions on Computer-Aided Design*, Vol. CAD-4, No. 3, pp. 208-219, Jul. 1985.

[8] Uzi Yoeli, “A Robust Channel Router,” *IEEE Transactions on Computer-Aided Design*, Vol. 10, No. 2, pp. 212-219, Feb. 1991.

[9] Kim Marriott and Peter J. Stuckey, *Programming with Constraints: An Introduction*, The MIT Press, 1998.

[10] http://www.jacop.eu

[11] Krzysztof Kuchcinski, “Constraints-Driven Scheduling and Resource Assignment,” *ACM Transactions on Design Automation of Electronic Systems*, Vol. 8, No. 3, pp. 355-383, Jul. 2003.

[12] Krzysztof Kuchcinski and Christophe Wolinski, “Global Approach to Assignment and Scheduling of Complex

Behaviors Based on HCDG and Constraint Programming," *Journal of Systems Architecture*, Vol. 49, pp. 489-503, Dec. 2003.

[13] I-Lun Tseng and Adam Postula, "Partitioning Parameterized 45-Degree Polygons with Constraint Programming," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 13, No. 3, pp. 52:1-52:29, Jul. 2008.

[14] Chung-Kuan Cheng and David N. Deutsch, "Improved Channel Routing by Via Minimization and Shifting," In *Proceedings of Design Automation Conference*, pp. 677-680, 1988.

[15] Tong Gao and C. L. Liu, "Minimum Crosstalk Channel Routing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 5, pp. 465-474, May 1996.

[16] Kuo-Chih Hsu, Yu-Chung Lin, Po-Xun Chiu, and Tsai-Ming Hsieh, "Minimum Crosstalk Channel Routing with Dogleg," In *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 73-76, 2000.

[17] Pralay Mitra, Nabin Ghoshal, and Rajat K. Pal, "A Graph Theoretic Approach to Minimize Total Wire Length in Channel Routing," In *Proceedings of IEEE TENCON - Conference on Convergent Technologies for Asia-Pacific Region*, pp. 414-418, 2003.

[18] Alvaro Ruiz-Andino and Jose J. Ruz, "Integration of Constraint Programming and Evolution Programs: Application to Channel Routing," *Methodology and Tools in Knowledge-Based Systems* (Lecture Notes in Computer Science), Springer, pp. 448-459, 1998.

[19] Nicholas C. Phillips, "Channel Routing by Constraint Logic," In *Proceedings of ACM Symposium on Applied Computing*, pp. 536-540, 1992.

[20] David N. Deutsch, "A 'Dogleg' Channel Router," In *Proceedings of Design Automation Conference*, pp. 425-433, 1976.