# Dogleg Channel Routing with Parallel Mixed Integer Linear Programming Solvers

**I-Lun Tseng[1], Yung-Wei Kao[1], Cheng-Yuan Chang[1], and Adam Postula[2]**
[1]Department of Computer Science and Engineering, Yuan Ze University, Taiwan
[2]School of Information Technology and Electrical Engineering, The University of Queensland, Australia

**Abstract -** *Channel routing is a type of problems arising in the detailed routing phase of VLSI physical design automation as well as in the design of printed circuit boards (PCBs). It has been known that channel routing problems can be formulated as constraint programming (CP) problems and thus CP solvers can be used to find solutions. In this article, we present a mixed integer linear programming (MILP) formulation to gridded dogleg channel routing problems. As a result, parallel MILP solvers, which have the ability to exploit the computational power of multi-core processors, can be used to find solutions. Experimental results show that high degrees of scalability can be achieved. Owing to the properties of MILP problems, it is possible to further shorten the execution time if a computer containing more processor cores is available.*

**Keywords:** Channel Routing, VLSI Physical Design Automation, Mixed Integer Linear Programming, Parallel Computing

## 1 Introduction

Channel routing is a type of problems arising in the detailed routing phase of VLSI physical design automation [2] as well as in the design of printed circuit boards (PCBs) [3]. Although channel routing has not been an active research field in recent years, the use of mixed integer linear programming (MILP) 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 [4].

In order to solve a channel routing problem, many routing algorithms generate horizontal and/or vertical constraints for the problem [5, 6]. For a channel routing problem containing cyclic vertical constraints, doglegs are required in order to complete the routing [6]. Since dogleg channel routing problems are NP-complete [7], many heuristic algorithms have been developed and proposed [6, 8, 9]. 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 an MILP problem. Consequently, an MILP solver can be used to find optimal solutions of the transformed problem. 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.

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 [8], minimizing the number of vias [10], minimizing the crosstalk [11, 12], and minimizing the total wire length [13]. Most 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 [11]).

This paper is organized as follows. In Section 2, we present a systematic approach which can be used to transform a constraint programming (CP) problem into an MILP problem. Section 3 discusses dogleg channel routing problems and the representations of their routing results. Left and right connection sets are presented in Section 4, followed by the MILP-based channel routing algorithm. Section 6 shows experimental results. Finally, conclusions are drawn in Section 7.

## 2 CP to MILP Transformation

Constraint programming (CP) [14] is a type of declarative programming paradigm since it allows users to specify a problem in terms of variables and constraints over those variables. After a problem has been specified, a CP solver can be used to find the solution(s) to the problem. CP solvers have been used in solving many difficult problems (such as optimization problems, scheduling and resource assignment problems [15, 16], and problems of partitioning parameterized polygons [17]), although the time complexity for solving those problems may not be polynomial.

Linear programming (LP) [18] also allows users to specify a problem in terms of variables and constraints. However, each of these constraints must be a linear equality or a linear inequality constraint. In addition, a linear objective function can exist and its value can be either minimized or maximized. If a problem can be expressed in this way, it is

This research was supported in part by the National Science Council of Taiwan under grants NSC-98-2221-E-155-053 and NSC-99-2221-E-155-088.

called an LP problem. If a problem can be expressed as an LP problem and all of the variables are required to be integers, then it is called an integer linear programming (ILP) problem. Furthermore, if a problem can be expressed as an LP problem and some of the variables are required to be integers, it is called a mixed integer linear programming (MILP) problem.

Over the past few years, technologies of MILP solvers have advanced. A number of commercial MILP solvers (e.g., CPLEX and Gurobi Optimizer) are now capable of exploiting computational power from multi-core processors. Therefore, by transforming a CP problem into an MILP problem, parallel MILP solvers can be used to solve the original problem. As a result, computational resources from multi-core processors can be highly utilized and the execution time can be reduced. This section describes how to transform a number of fundamental types of linear and/or logical constraints, which reside in a CP problem, into linear inequalities.

In this section, symbols $A$, $B$, $C$, and $D$ are used; each of them can be a constant, a variable, or a linear expression. Also, we assume that values for all of the constants, variables, and linear expressions are integers.

## 2.1 Logical-AND Constraints

Since a CP problem can have many constraints and a solution to the CP problem must satisfy all of the constraints, the relation between each pair of these constraints is considered as conjunction (logical-AND). In a CP solver, therefore, when we need to specify a set of conjunctive constraints, usually we only need to impose these constraints one by one. Similarly, in an MILP solver, conjunctive constraints can be specified one by one.

## 2.2 Equality Constraints

For a constraint whose type is in the form of ($A = B$), it can be transformed into two conjunctive inequalities ($A \leq B$) and ($B \leq A$); each of $A$ and $B$ can be a constant, a variable, or a linear expression. As an example, the linear constraint ($3*A - 2*B = 5*C + 26$) can be transformed into two linear inequalities.

## 2.3 Less-Than and Greater-Than Constraints

A constraint which is in the form of ($A < B$) can be converted into the inequality ($A \leq B - 1$) if values of both $A$ and $B$ are integers. Likewise, a constraint which is in the form of ($A > B$) can be transformed into the inequality ($B \leq A - 1$).

## 2.4 Logical-OR Constraints

In order to convert a constraint which contains a logical-OR operation into inequalities, an extra binary variable, whose value is either 0 or 1, can be used; the binary variable can also be called a 0-1 integer. For example, a constraint which is in the form of (($A \leq B$) OR ($C \leq D$)) can be transformed into the following two conjunctive inequalities:

- $A \leq B + \text{M1}*BV$
- $C \leq D + \text{M2}*(1-BV)$

In the above inequalities, $BV$ is a binary variable. Also, both of M1 and M2 are constants whose values must be sufficiently large so that the second inequality is redundant if $BV$=0 and the first inequality is redundant if $BV$=1. Similarly, a constraint which contains two or more logical-OR operations can be converted into a number of inequalities by using two or more binary variables, as one of the examples shown in Section 2.6.

## 2.5 Not-equal Constraints

A not-equal constraint which is in the form of ($A \neq B$) can be converted into two conjunctive inequalities as the steps shown below:

$A \neq B$
⇨ ( $A < B$ ) OR ( $B < A$ )
⇨ ( $A \leq B - 1$ ) OR ( $B \leq A - 1$ ) [Section 2.3]
⇨ ( $A \leq B - 1 + \text{M1}*BV$ ) AND ( $B \leq A - 1 + \text{M2}*(1-BV)$ ) [Section 2.4]

As described in Section 2.4, $BV$ is a binary variable, and the values of M1 and M2 must be sufficiently large.

## 2.6 If-Then-Else and If-Then Constraints

This subsection shows two examples of conditional constraints and how they can be transformed into linear inequalities. Other types of conditional constraints can be transformed by using similar techniques. The first example is the following If-Then-Else constraint:

- IF ($A \leq B$) THEN ($C \leq D$) ELSE ($E \leq F$)

The constraint can be split into two disjunctive (logical-OR) constraints as shown below:

- ( ($A \leq B$) AND ($C \leq D$) )
- ( ($A > B$) AND ($E \leq F$) )

We can then add a binary variable, $BV$, in order to further transform these two disjunctive constraints. The final transformed conjunctive inequalities are listed below:

- $\text{A} \leq \text{B} + \text{M1}*BV$
- $\text{C} \leq \text{D} + \text{M2}*BV$
- $\text{B} \leq \text{A} - 1 + \text{M3}*(1-BV)$
- $\text{E} \leq \text{F} + \text{M4}*(1-BV)$

All of M1, M2, M3, and M4 are integer constants and their values must be sufficiently large. Therefore, ($A \leq B$) and ($C \leq D$) must hold when $BV$=0. Also, ($A > B$) and ($E \leq F$) must hold when $BV$=1.

The second example is the following If-Then constraint, which is more complicated than the previous example but can also be converted into linear inequalities:

- IF ($A < B$) THEN ($C = D$) OR ($E < F$ AND $G < H$) OR ($I < J$ AND $K < L$)

Table I. Conjunctive Inequalities Transformed from the Constraint "IF ($A < B$) THEN ($C = D$) OR ($E < F$ AND $G < H$) OR ($I < J$ AND $K < L$)"

| Constraint ID | Constraint | Values of Binary Variables for Activating Corresponding Constraints |
|---|---|---|
| C-1 | $A \leq B - 1 + \text{M1}*B1 + \text{M2}*B2$ | If ($B1 = 0$) and ($B2 = 0$), then ($A < B$) and ($C = D$). |
| C-2 | $C \leq D + \text{M3}*B1 + \text{M4}*B2$ | |
| C-3 | $D \leq C + \text{M5}*B1 + \text{M6}*B2$ | |
| C-4 | $A \leq B - 1 + \text{M7}*B1 + \text{M8}*(1-B2)$ | If ($B1 = 0$) and ($B2 = 1$), then ($A < B$) and ($E < F$) and ($G < H$). |
| C-5 | $E \leq F - 1 + \text{M9}*B1 + \text{M10}*(1-B2)$ | |
| C-6 | $G \leq H - 1 + \text{M11}*B1 + \text{M12}*(1-B2)$ | |
| C-7 | $A \leq B - 1 + \text{M13}*(1-B1) + \text{M14}*B2$ | If ($B1 = 1$) and ($B2 = 0$), then ($A < B$) and ($I < J$) and ($K < L$). |
| C-8 | $I \leq J - 1 + \text{M15}*(1-B1) + \text{M16}*B2$ | |
| C-9 | $K \leq L - 1 + \text{M17}*(1-B1) + \text{M18}*B2$ | |
| C-10 | $B \leq A + \text{M19}*(1-B1) + \text{M20}*(1-B2)$ | If ($B1 = 1$) and ($B2 = 1$), then ($A \geq B$). |

This constraint can be divided into four disjunctive (logical-OR) constraints as listed below:

- ($A < B$) AND ($C = D$)
- ($A < B$) AND ($E < F$) AND ($G < H$)
- ($A < B$) AND ($I < J$) AND ($K < L$)
- ($A \geq B$)

We can then add two binary variables, *B1* and *B2*, in order to further transform these disjunctive constraints. The transformation result, which contains ten conjunctive inequalities, is shown in Table I.

# 3 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 [19] 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 [20], 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 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 Figure 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]

# 4 Left and Right Connection Sets

A channel routing problem can have a left connection set (LCS) and/or a right connection set (RCS) [4]. The left (right) connection set refers to the set of nets which enters/exits the channel from the left (right). To tackle these types of nets, we

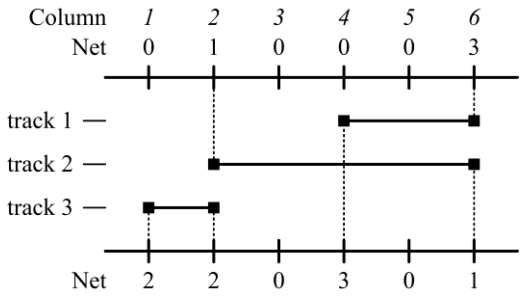


**Figure 1.** A (gridded) channel routing problem and one of its solutions without using doglegs

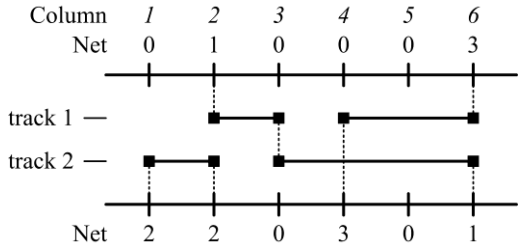


**Figure 2.** A solution to the channel routing problem (shown in Figure 1) with the use of doglegs

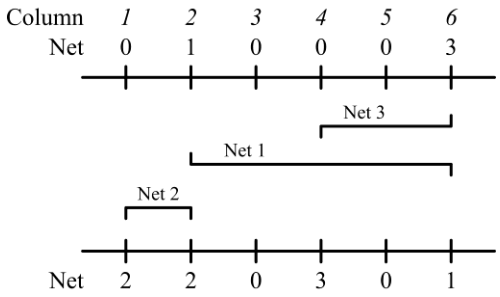


**Figure 3.** Horizontal span of each net

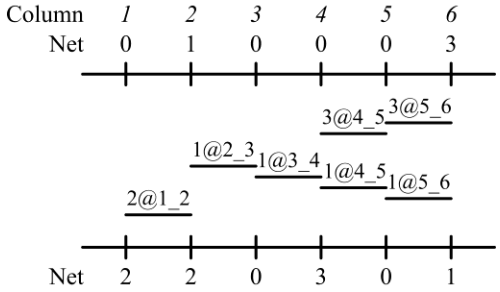


**Figure 4.** Horizontal wire fragments and their representations

add a new column at the left of the channel if the LCS exists, and add a new column at the right of the channel if the RCS exists. Also, new horizontal fragments must be created. For the newly created columns, their corresponding vertical constraints [21] do not need to be generated. However, horizontal constraints [21] for the newly created fragments are required. Figure 5 illustrates an example of a channel routing problem with the existence of LCS and RCS; the problem can be transformed into the one shown in Figure 6.

# 5 The Algorithm

The methodology of formulating gridded dogleg channel routing problems as CP problems has been presented in [21]. Additionally, by using the methodology presented in Section 2, horizontal and vertical constraints presented in [21] can be transformed into conjunctive linear inequalities. As a result, a dogleg channel routing problem formulated as a CP problem can be transformed into an MILP problem. The final MILP-based dogleg channel routing algorithm is shown in Figure 7. To the best of our knowledge, our approach is the first in the literature to model the problems of gridded dogleg channel routing with via minimization as MILP problems. Moreover, our approach does not require a given initial routing result as the input.

In our MILP-based channel routing algorithm, a gridded dogleg channel routing problem is transformed into an MILP problem and then solved by an MILP solver. Since *channel density* is the minimum number of tracks required in order to solve a two-layer channel routing problem [9], 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*] initally. In the algorithm, moreover, if the transformed MILP 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; it is also possible to override the value of *Max* manually. The algorithm stops when a solution has been found.

# 6 Experimental Results

The proposed algorithm has been implemented in Java programming language. Also, a number of testcases have been used to verify the correctness and to measure the performance of our MILP-based channel routing program. Information for some of those testcases is shown in Table II, and the experimental results are shown in Table III. Note that Table III shows results of executing our programs with the via minimization function turned on. Testcases and routing results of “Figure 2” and “Figure 9” can be seen in Figure 2 and Figure 9, respectively. Figures 8 and 9 illustrate the same routing problem with different routing results; one is with the via minimization function turned off and the other turned on. Testcases “Phillips_14” and “Phillips_16” are from [1]; the former has 14 columns and the latter has 16 columns. The testcase named “Deutsch” is the notable Deutsch’s difficult example [4, 19].

To compare the efficiency of the MILP-based channel routing program with the CP-based channel routing program [21], we used JaCoP version 2.4.1 (released in 2009) as the CP solver. In our MILP-based program, we could choose to

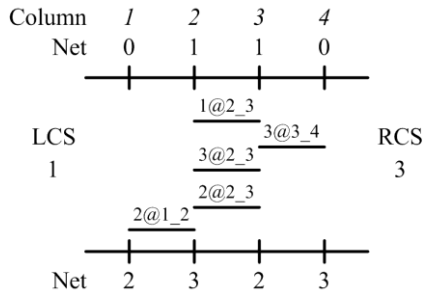


**Figure 5.** A channel routing problem with the existence of LCS and RCS

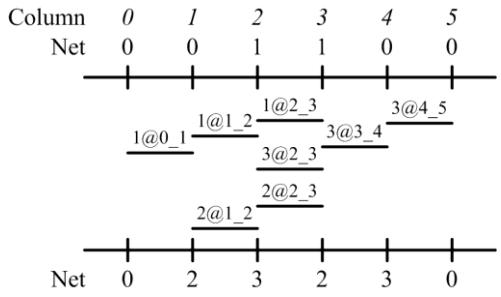


**Figure 6.** A new channel routing problem transformed from the channel routing problem shown in Figure 5

use CPLEX version 10.2 (released in 2007), Gurobi version 2.0.2 (released in 2009), or CPLEX version 12.1 (released in 2009) as the MILP solver. In our experiments, all of the CP and MILP solvers were run on a workstation which had an Intel Q9550 CPU (2.83 GHz, Quad Core) and 8 GB of RAM, except that CPLEX v.10.2, which was run on a server having an Intel Xeon E5345 CPU (2.33 GHz, Quad Core) and 32 GB of RAM. Also, we extracted some experimental results from [1] and put them in Table III; a machine with a 12 MHz CPU was used to generate the original results. Please note that Phillips did not implement the via minimization function.

In [1], the Phillips_16 case with 8 tracks could not be solved within 40 hours of execution time and no result was generated. By using our MILP-based channel routing algorithm with CPLEX v.12.1, we proved that the Phillips_16 case with 8 tracks was infeasible within one minute of time. From the experimental results shown in Table III, we found that CPLEX v.12.1 performed efficiently in identifying infeasible problems, while Gurobi v.2.0.2 performed relatively well in solving large and complex feasible problems. In addition, the performance of JaCoP v.2.4.1 was relatively unstable, but it solved some large problems rapidly.

Although our approach of using MILP solvers 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 case in Table III, suboptimal results can be generated in exchange for significantly reduced execution time. Moreover, as will be shown in the next paragraph, if more processor cores are available, the execution time can be reduced.

Table IV shows the elapsed time of running MILP-based channel routing with different parallel MILP solvers and with different number of threads. While we were running testcases by using four threads, it was common to see that the CPU utilization reached around 400% (by using Linux's "top" command), which meant that all of the four CPU cores had participated in the computation. In general, as shown in Table IV, if more CPU cores are available, the execution time can be reduced. In the results of solving the Deutsch's difficult problem with 25 tracks, it was interesting to note that nearly 8x speedup had been achieved when the number of threads was increased from one to four. That is because, while using a parallel solver in solving an optimization problem, one thread can pass information to other threads when a feasible solution has been found by the thread. As a result, many unnecessary computations can be eliminated.

**Algorithm** MILP-CHANNELROUTER(*TR*, *BR*, *LCS*, *RCS*, *VM*)

*Input*. The input contains the description of 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, (3) *LCS*, and (4) *RCS*. Also, a Boolean variable *VM* is used in order to control the activation of the via minimization function.

*Output*. The output contains a permutation of horizontal wire fragments, from which a solution to the input channel routing problem can be constructed.

1. $D \leftarrow$ the channel density of the input channel routing problem
2. $Max \leftarrow D$
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.
6. Generate vertical constraints for each column.
7. Generate constraints for the left and right connection sets (Section 4).
8. **if** *VM* equals TRUE **then**
9. Generate variables and constraints for minimizing the number of vias.
10. Specify generated variables and constraints via API functions of an MILP solver.
11. Invoke the MILP solver to solve the specified MILP problem. When a solution has been found, report the solution and then exit the algorithm.
12. $Max \leftarrow Max + 1$
13. } **while** (the MILP solver has not found a solution)

**Figure 7.** The MILP-based dogleg channel routing algorithm

# 7 Conclusion

In order to exploit the computational power of multi-core processors, we proposed a systematic approach which can be used to transform a CP problem containing linear and logical constraints into an MILP problem. The approach is capable of transforming a gridded dogleg channel routing problem with via minimization, formulated as a CP problem, into an MILP problem. Experimental results have shown that high degrees

Table II. Information of Channel Routing Problems

| Testcase | # of nets | # of columns | channel density | min. # of tracks |
|---|---|---|---|---|
| Figure 2 | 3 | 6 | 2 | 2 |
| Figure 9 | 5 | 9 | 5 | 6 |
| Phillips_14 | 10 | 14 | 4 | 5 |
| Phillips_16 | 11 | 16 | 8 | 9 |
| Deutsch | 52 | 156 | 19 | 19 |

# of nets: the number of nets in the testcase; # of columns: the number of columns in the testcase; channel density: the channel density of the testcase; min. # of tracks: the minimum number of tracks required for solving the channel routing problem.

Table III. Experimental Results of CP-based and MILP-based Channel Routing

| Testcase | # of tracks used | Phillips [1] | JaCoP v.2.4.1 1 thread | CPLEX v.10.2 1 thread | Gurobi v.2.0.2 4 threads | CPLEX v.12.1 4 threads |
|---|---|---|---|---|---|---|
| Figure 2 | 2 | N.A. | 0.14 sec. | **0.00 sec.** | **0.00 sec.** | **0.00 sec.** |
| Figure 9 | 6 | N.A. | **0.37 sec.** | 1.96 sec. | 0.48 sec. | 0.81 sec. |
| | 5 | N.A. | 154766 sec. (infeasible) | 0.24 sec. (infeasible) | 0.32 sec. (infeasible) | **0.21 sec.** (infeasible) |
| Phillips_14 | 6 | 0.28 sec. | 253.55 sec. | 0.32 sec. | 0.27 sec. | **0.20 sec.** |
| | 5 | < 1.37 sec. | 0.36 sec. | 0.08 sec. | **0.04 sec.** | 0.09 sec. |
| | 4 | < 1.37 sec. (infeasible) | 0.19 sec. (infeasible) | **0.00 sec.** (infeasible) | **0.00 sec.** (infeasible) | **0.00 sec.** (infeasible) |
| Phillips_16 | 10 | N.A. | > 7 days (unfinished) | 8.49 sec. | **0.71 sec.** | 0.92 sec. |
| | 9 | 0.39 sec. | > 7 days (unfinished) | 27.37 sec. | **0.67 sec.** | 0.93 sec. |
| | 8 | > 40 hrs. (unfinished) | > 7 days (unfinished) | 54646.89 sec. (infeasible) | 520.23 sec. (infeasible) | **55.34 sec.** (infeasible) |
| Deutsch | 25 | N.A. | **4.18 sec.** | > 3 days (unfinished) | 3451.79 sec. | > 7 days (unfinished) |
| | 24 | N.A. | **5.10 sec.** | > 3 days (unfinished) | 174270.72 sec. | > 7 days (unfinished) |
| | 23 | N.A. | > 7 days (unfinished) | > 3 days (unfinished) | **136266.61 sec.** | > 7 days (unfinished) |

# of tracks used: the number of tracks used by our CP-based or MILP-based channel routing program for solving the problem (the value was assigned by setting the *Max* manually).

of scalability can be achieved by using parallel MILP solvers. Our approach can be further extended to consider crosstalk and total wire length. Although the experimental results show that the execution 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 further improved with the advance of mixed integer linear programming technologies as well as the advance of multi-core processors.

# References


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

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

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

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

[5] 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, July, 1990.

[6] 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, January, 1982.

[7] Thomas G. Szymanski, "Dogleg Channel Routing is NP-Complete," *IEEE Transactions on Computer-Aided Design,* vol. CAD-4, no. 1, pp. 31-41, 1985.

[8] 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, 1985.

[9] Uzi Yoeli, "A Robust Channel Router," *IEEE Transactions on Computer-Aided Design,* vol. 10, no. 2, pp. 212-219, February, 1991.

[10] 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.

[11] 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.

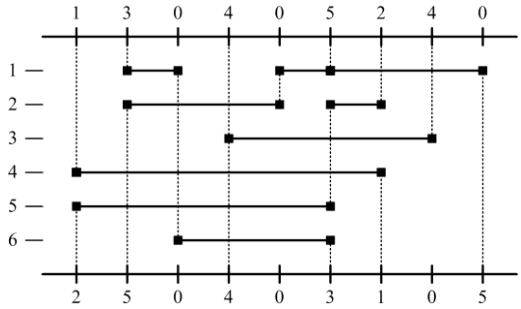


**Figure 8**. A minimum-track solution to a dogleg channel routing problem without via minimization (total number of vias = 17)

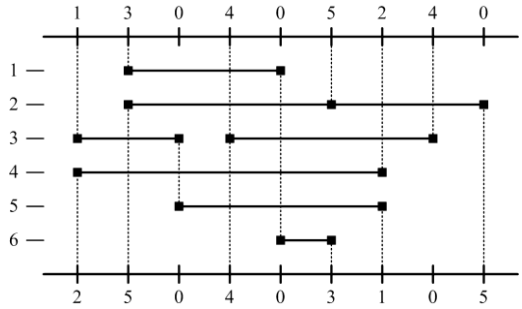


**Figure 9**. A minimum-track solution to the dogleg channel routing problem (shown in Figure 8) with the via minimization function turned on (total number of vias = 15)

Table IV. Executing MILP-Based Channel Routing with Different Number of Threads

| Testcase | # of tracks used | CPLEX v.12.1 | | | Gurobi v.2.0.2 | |
|---|---|---|---|---|---|---|
| | | 1 thread | 2 threads | 4 threads | 1 thread | 4 threads |
| Phillips_16 | 8 | 209.42 sec. | 127.87 sec. | 55.34 sec. | 1708.20 sec. | 520.23 sec. |
| Deutsch | 25 | > 3 days (unfinished) | > 3 days (unfinished) | > 3 days (unfinished) | 27410.08 sec. | 3451.79 sec. |


[12] 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.

[13] Pralay Mitra, Nabin Ghoshal, and Rajat K. Pal, "A Graph Theoretic Approach to Minimize Total Wire Length in Channel Routing," In *Proceedings of IEEE Region 10 Conference (TENCON)*, pp. 414-418, 2003.

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

[15] Krzysztof Kuchcinski, "Constraints-Driven Scheduling and Resource Assignment," *ACM Transactions on Design Automation of Electronic Systems,* vol. 8, no. 3, pp. 355-383, 2003.

[16] 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, 2003.

[17] 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, July, 2008.

[18] Wayne L. Winston, and Munirpallam Venkataramanan, *Introduction to Mathematical Programming*, Thomson Learning, Inc., 2003.

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

[20] 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, January 1982.

[21] I-Lun Tseng, Huan-Wen Chen, Che-I Lee, and Adam Postula, "Constraint-Based Dogleg Channel Routing with Via Minimization," In *Proceedings of International Conference on Artificial Intelligence (ICAI)*, pp. 666-672, 2010.