Mathematical Methods of Operations Research manuscript No.
(will be inserted by the editor)

Via Minimization in VLSI Chip Design

Application of a Planar Max-Cut Algorithm

Frauke Liers · Tim Nieberg · Gregor Pardella

Received: date / Accepted: date

Abstract The design of *very large scale integrated (VLSI)* chips is an exciting area of applied discrete mathematics. Due to the intractability of the majority of the problems, and also due to the huge instance sizes, the design process is decomposed into various sub-problems. In this paper, for a given detailed routing solution, we revisit the assignment of layers to net segments. For connected metalized nets, a layer change is accomplished by a vertical interconnection area (*via* for short). We seek to minimize the use of these vias as vias not only reduce the electrical reliability and performance of the chip, but also decrease the manufacturing yield substantially. In the general case, the via minimization problem is NP-hard. However, it is known that the two layer via minimization problem can be solved as a maximum cut problem on a planar graph which is a polynomial task. The focus of this paper is to use this approach for modern real-world chips. From the roughly two dozen wiring layers present, we take two adjacent ones for the via minimization. As a core-routine, we use a fast maximum cut algorithm on planar graphs. For being able to use the solutions in practice, we integrate practically relevant design rule constraints at the expense of potentially using further vias. Thus, our solution satisfies the additional constraints present in actual current designs. The computational results show that our implementation is fast on real-world instances as it usually computes a solution within a few minutes CPU time only. Moreover, often a considerable amount of vias can be saved.

Keywords VLSI design · maximum cut · via minimization · manufacturing yield

Financial support from the German Science Foundation is acknowledged under contract Li 1675/1.

F. Liers
Universität zu Köln, Institut für Informatik, Pohligstraße 1, 50969 Köln, Germany
E-mail: liers@informatik.uni-koeln.de

T. Nieberg
Research Institute for Discrete Mathematics, University of Bonn, Lennéstr. 2, 53113 Bonn, Germany
E-mail: nieberg@or.uni-bonn.de

G. Pardella
Universität zu Köln, Institut für Informatik, Pohligstraße 1, 50969 Köln, Germany
E-mail: pardella@informatik.uni-koeln.de

1 Introduction

In the VLSI design process, routing is a complex and hard problem both in theory and practice [1]. The output represents the later metalized areas on the chip area, thus making logical connections of the overall circuit electrically connected.

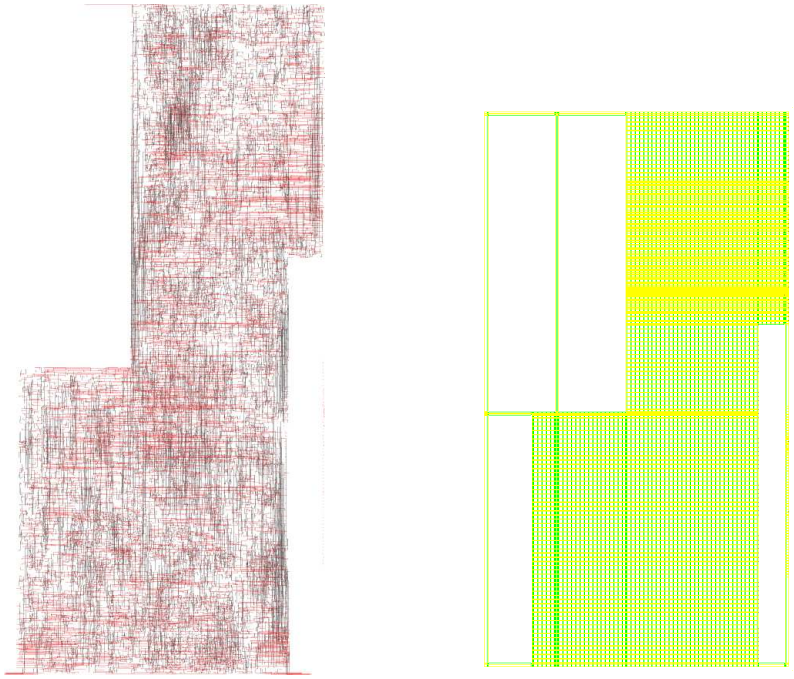
In an abstract view, modern chips consist of several dozens of *layers*, i.e. conductive material of the chip, on which *nets* are routed that connect *pins* (or *terminals*) to make them electrically equivalent. In other words, nets are connections between chip elements such as logical modules and circuits, and of course different nets have to be geometrically/electrically disjoint. Nets are built of *wire segments* that represent the later metalized rectangular areas on each wiring plane. The connection point where two or more segments meet is called *junction*. Its *degree* is the number of incident segments. If two or more adjacent segments of a net are placed on different layers, a *vertical interconnect access*, or short *via*, is necessary.

Vias not only reduce the reliability and electrical performance of the chip (vias have higher resistance), but also have a major impact on the manufacturing yield: the number of vias is inversely related to the yield because a chip with more vias has a smaller probability of being fabricated correctly. Therefore, it is desirable to minimize the number of vias introduced in VLSI routing, and we concentrate here on the case with only two adjacent layers as an important special case in a practical setting.

In practice, however, via minimization is often either neglected or de-emphasized in routing tools, and comes as an afterthought problem. Furthermore, most detailed routing solutions work in a sequential fashion. In fact, it is common design practice to assign all vertical wire segments to one layer and all horizontal wire segments to the other. Hence, a large number of layer changes might be introduced to interconnect the wire segments on different layers. In Figure 1 we see the detailed routing of a real-world chip with respect to two relevant layers, see Figure 1a. In Figure 1b the corresponding power grid is shown. The latter correspond to *blockages* in the layer assignment problem. A segment cannot be placed on a layer if it intersects the power supply grid or any other shape that is considered as blockage on it.

Given an initial wiring consisting of a set of wire segments, the problem to assign wire segments to layers such that the topology and the logical connections are maintained and the number of vias required is minimized, is the so-called *constrained via minimization* problem. We call a wiring solution given as input where the layers or a subset thereof are collapsed into a single plane, therefore also neglecting all vias, a *transient routing*. The problem at hand is thus also referred to as *layer assignment* since it deals with assigning layers to all segments. We say that a *feasible layer assignment* is a *realizable layer assignment* in two layers if it respects all additional (design rule) constraints.

The constrained via minimization problem for two layers originated in the pioneering work of Hashimoto and Stevens [16] 1971. However only 2-way junctions were allowed in their model. In 1980, Kajitani [17] proposed a polynomial-time algorithm for a special case of the problem. An integer-programming formulation was introduced in [11]. In general, the via minimization problem is NP-hard [22, 9]. For certain restricted cases, polynomial-time solutions are possible. Indeed, in 1983, Chen



(a) (b)

Fig. 1: A real-world chip. (a) Wire segments after routing. Red segments are placed on one layer while black segments are placed on the other layer. (b) Power grid which yields blockages that have to be taken care of when assigning layers to the segments.

et al. [8] presented a polynomial-time algorithm for grid-based layouts which gives optimum results when the maximum junction degree is limited to three and heuristic results otherwise. However, they restrict vias to lie at junctions which already existed in the wiring. This preludes the placement of a via on any straight-line segment, thus, limiting the applicability of their approach.

Independently, Pinter [23] proposed a polynomial-time algorithm in the case that at most 3-way junctions occur. The algorithm is based on determining a partition of the vertex set in an appropriate planar conflict graph such that the sum of edge weights of edges connecting vertices in different partitions is maximum. The latter is known as *maximum cut problem* on planar graphs. Chang and Du [6] developed a heuristic algorithm that can handle junctions of any degree. It however does not guarantee optimum solutions. Naclerio et al. [21] developed an algorithm with the same time complexity as the method proposed in [8], but does not require the layout to be grid based and does not restrict the placement of vias. Their algorithm yields optimum results when the junction degree is limited to three and heuristic results otherwise. Barahona et al. [3] and Grötschel et al. [15] followed the approaches in [8, 23] and proposed a branch-and-cut algorithm in which practical constraints, like pin preassignment and layer preferences, were included.

Chang and Du [7] reduced the three coloring problem on planar graphs to the constrained via minimization problem on three layers and proved that the latter is NP-hard. In 1997, Chang and Cong [5] extended Pinter's approach to k layers and presented a heuristic method to obtain a k-cut solution yielding a layer assignment. Chou and Lin [10] used the relation between the k-layer constrained via minimization problem and the constrained k-way graph partitioning problem and proposed a simulated annealing approach which showed good results on practical instances. Recently, Fouilhoux and Mahjoub [12, 13] reduced the k-layer via minimization problem with arbitrary junction degree to the k-partite induced subgraph problem. They gave an exact integer programming formulation and proposed additional constraints for pin preassignment, stacked vias, etc.

However, as modern chips consists of a very large number of nets, it is usually not possible to use branch-and-cut approaches for real-world chips. We focus here on applying the polynomial-time approach of [8, 23]. For determining maximum cuts in planar graphs, we use the exact polynomial-time algorithm presented in [19] as a core routine. The latter works as follows. First, the dual of the planar graph is constructed. By including a limited amount of artificial nodes and edges, the dual is extended such that the degree of each vertex is at most four. Then, each vertex is replaced by a complete graph on four nodes. An optimum perfect matching is calculated on this extended dual graph. The optimum matching is then used for determining an optimum cut in the original input graph. This approach can determine maximum cuts in planar graphs with more than 10^6 nodes within short time. Furthermore, implementing the method is a straight-forward task, when using a publicly available implementation for perfect matchings. As our planar maximum-cut algorithm has no knowledge of chip design rules or practical constraints such as blockage or pin preferences, we satisfy them by potentially investing further vias. Therefore, our polynomial-time approach yields solutions that might need more vias than actually required for a given wiring. This does not come as a surprise as these further constraints make the via minimization problem NP-hard.

In the next section we briefly recall a transformation of the layer assignment problem to a max-cut problem on an appropriate conflict graph following [8, 23, 3, 15]. The placement of vias is not restricted to junctions. Furthermore, they can be placed always on junctions or can be placed on straight line segments when possible. Additional side constraints are introduced one by one in Section 3. We explain how to integrate them into the polynomial approach. The method can be used for any chip geometry. Then results for real world chips are presented and discussed in Section 4. Finally, we give a conclusion in Section 5.

2 Transformation to a Max-Cut Problem on a Planar Graph

In the following we review the transformation of the via minimization problem to a max-cut problem on an appropriate *conflict graph*. We follow the approaches proposed in [8, 23, 3, 15]. Suppose we are given a transient routing of the chip. We assume that at most 3-way junctions occur. Further, no net connects more than three

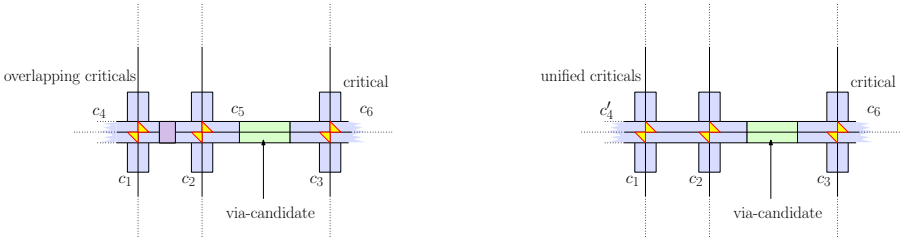


Fig. 2: Segment intersections. Within each critical interval vias are not allowed to be placed. All non-critical parts are via-candidates, vias may be placed here only. (a) Criticals c_4 and c_5 overlap. The counter-part to c_3 is c_6 and vice versa. (b) The result of a unify operation on c_4 and c_5 is critical c'_4 with its counter-parts c_1 and c_2. The counter-part for both c_1 and c_2 is now c'_4.

pins. Later we will drop these restrictions and discuss in Section 3 how to integrate additional practical constraints such as design rules efficiently.

We partition the segments into *critical* and *via-candidate* intervals, see Figure 2a. In the following we say *criticals* and *via-candidates* for short to address critical resp. via-candidate intervals. Criticals are defined by an intersection of at least two segments. These segments (or parts of them) have to be placed on different layers. It is easy to see that the crossing of more than two segments (in the same point) imply that the routing cannot be realized in two layers. Moreover, as vias need space, the crossing defines the middle of an interval such that a via can be placed at the interval borders without introducing a short-circuit. We refer to these intervals as criticals. On the other hand, via-candidates are all regions of a net that are not critical. Vias can be placed on via-candidates only. By applying a sweep line algorithm (such as proposed in [24,4]) we find all criticals. Each crossing of two segments implies two criticals, one on each participating segment. Thus, each critical implies at least another 'counter-part' critical. Next, we search for overlapping criticals because they do not leave enough space for vias to be placed in between. Overlapping criticals are unified to a larger critical. A direct consequence is that there may be more critical counter-parts for one critical, see Figure 2b.

We have partitioned the chip into criticals and via-candidates. Now we express the conflicting segments by a *conflict graph* $G = (V, E)$. If there are no conflicts the chip can be directly realized in two layers. Otherwise, each critical defines a vertex in G. We introduce two kinds of edges, *conflict edges* and *free edges*. The former are those edges that connect a critical with its counter-part(s). Note that these counter-part(s) have to be placed on layers different than that of the critical. Two vertices are joined by a free edge if the two corresponding criticals are incident to the same via-candidate.

If each segment incident to a 3-way junction leads to a critical, we mutually connect the three of them by so-called *half-free edges*, see Figure 3.

So $E = A \cup B$, where A is the set of conflict edges and B the set of free edges. It is easy to see that the subgraph $G_A = (V, A)$ induced by the set of conflict edges is bi-

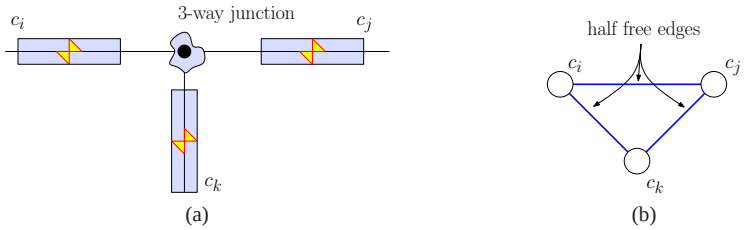


Fig. 3: Three criticals connected by half-free edges as they are incident to a 3-way junction.

partite. Otherwise, there exist conflicts that cannot be resolved in two layers. Furthermore, G_A partitions into connected bipartite components with vertex sets $V_i \subseteq V$.

On the other hand, the subgraph $G_B = (V, B)$ induced by the set of free edges is a planar graph. If not, some free edges cross which contradicts the definition of free edges that correspond to via-candidates.

Now we can model the via minimization problem as a cut problem on G. A cut that contains all conflict edges and minimizes the number of free edges in the cut corresponds to a realizable layer assignment with a minimum number of vias. Note that half-free edges only count half. The layer assignment of each bipartite component can be determined if one of its vertices is fixed to a layer. Hence, we search for a cut that minimizes the number of free edges in the cut when we fix a partition for one representative vertex in each component. Consider a reduced conflict graph $G_R = (V_R, B_R)$ that is built by shrinking each bipartite component V_i in G to one vertex v_i in this component. We call this vertex the *representative* for component V_i. Hence $V_R = \{v_i \mid v_i \text{ is the representative for component } V_i\}$. Two representatives v_i and v_j are adjacent if and only if there exists a free edge connecting a vertex in component V_i to a vertex in V_j. Note that loops may occur. Hence $B_R = \{(v_i, v_j) \mid \exists\, e = (v_{i_k}, v_{j_l})\ v_{i_k} \in V_i, v_{j_l} \in V_j\}$.

It is easy to see that a layer assignment corresponds to a vertex partition Q and $V_R \setminus Q$ in G_R. By appropriately assigning weights to the edges in the graph, the maximum cut determines a layer assignment with the minimum number of vias.

Denote by n_S the number of segments. By applying a sweep line algorithm to all nets of a chip, all intersections can be identified in time $\mathcal{O}((n_S + I) \log n_S)$, where I denotes the number of intersections. The described transformation to the conflict graph and afterwards the construction of the reduced conflict graph can be done in linear time. Finding the optimum max-cut on $G_R = (V_R, B_R)$ (the most time consuming step) can be done in time $\mathcal{O}(|V_R|^{\frac{3}{2}} \log |V_R|)$, for example with the algorithm proposed in [19]. Thus, the asymptotic running time can be expressed as $\mathcal{O}((n_S + I) \log n_S + |V_R|^{\frac{3}{2}} \log |V_R|)$.

3 Satisfying Practical Constraints

In theory, most of the practical constraints drastically increase the complexity of the problem, resulting in NP-hard tasks. Nevertheless, we show how to integrate them

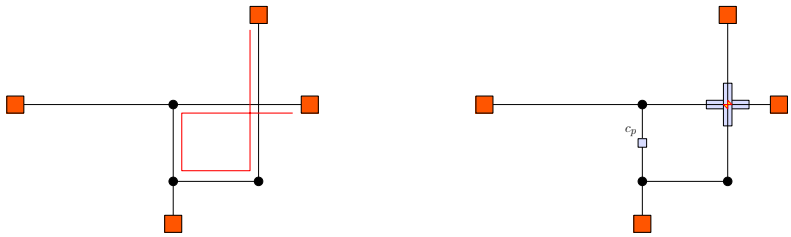


Fig. 4: Net connecting more than three pins. For realizing the latter, we allow for consecutive 3-way junctions. (a) In red: two pins connected by a path that passes two 3-way junctions. Note there is an intersection yielding two criticals. These are not only connected by a conflict edge but also adjacent to the same via-candidate which comprises two 3-way junctions. (b) Placed criticals. We introduce one point critical c_p.

in the basic via minimization algorithm from Section 2. in an efficient way, at the expense of additional vias.

3.1 Nets Connecting Four or more Pins

If we allow at most 3-way junctions then nets connecting more than three pins are possible if and only if there are at least two consecutive 3-way junctions on a path between two pins, see Figure 4a. Now, the subgraph G_B of the conflict graph $G = (V, A \cup B)$ induced by the set of free edges may not be planar anymore. To overcome this situation we introduce *point criticals*. These criticals are artificial and do not correspond to a real intersection of segments. We place them on each segment incident to a 3-way junction if (a) the segment does not already contain a critical and (b) it is part of a path connecting two criticals which passes two 3-way junctions. Everything else stays the same. Point criticals handle the possibility to place vias on one 3-way junction in case more than one 3-way junction lies between a pair of adjacent criticals. For the situation in Figure 4a we place a point critical c_p on the segment incident to the two 3-way junctions as shown in Figure 4b. The reduced conflict graph for this example consists of two vertices connected by an edge, and one via is needed. It is easy to see that we still have an exact polynomial approach for the via minimization problem when we use point criticals.

3.2 Minimum Distance

In a feasible routing solution, vias and wire segments not only have to be disjoint, but also need to obey certain minimum distance requirements (see Figure 5). In order to forbid the placement on such segments, we introduce *pseudo criticals*. We say that segments that are routed too close to each other are in *critical distance*. For dealing with overlapping segments, we extend the definition of criticals in a natural way. Hence, overlapping segments are handled as intersections. Pseudo criticals do not have critical counter-parts as they do not correspond to an intersection, and thus do

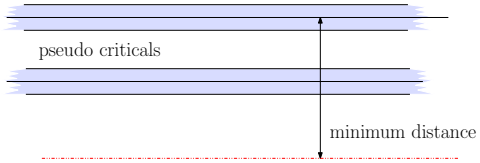


Fig. 5: Two segments running in parallel within minimum distance. We introduce pseudo criticals to ensure that no via is placed there. Otherwise a short circuit may be introduced. The minimum distance is indicated by the red dashed dotted line.

not have a layer preference. Hence we can independently place them on layers. Note that due to unifying operations they may become critical counter-parts. Again other construction rules are not changed. Algorithmically we extend the sweep line part from Section 2 to find segments within a critical distance. We do not only check for intersections of active segments but also run a test for segments in critical distance. This can be accomplished in the same running time as $\mathcal{O}((n_S + I) \log n_S)$, where I now denotes the number of intersections and segments within minimum distance. No further changes are needed. In a postprocessing step the best layer for those pseudo criticals is decided.

3.3 4-way Junctions

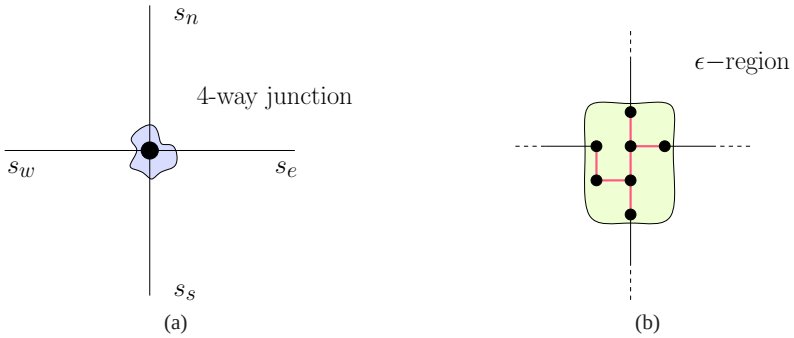


Fig. 6: (a) A 4-way junction. If such a junction occurs the net connects at least four pins. Allowing 4-way junctions makes the via minimization problem already NP-hard as the corresponding reduced conflict graph may be non-planar. (b) In a preprocessing step every 4-way junction is transformed to a chain of 3-way junctions within an ϵ-region. Each new introduced segment within the ϵ-region is critical. This transformation allows for not only 4-way junctions but for up to 8-way junctions.

Naclerio et al. [22] and Choi et al. [9] have shown that the two layer via minimization problem is NP-hard if one allows 4-way junctions. They reduced the vertex-deletion graph bipartization problem to the via minimization problem. Nowadays 4-way junctions are used more frequently in chip layouts. As we cannot expect to be able to introduce them into an exact polynomial time approach, we propose the

following transformation of a 4-way junction to a chain of 3-way junctions. Note that this is a heuristic way of modeling 4-way junctions where potentially additional vias need to be used. Nevertheless, we obtain good heuristic solutions in polynomial time. A layer assignment on the latter instance can then be mapped to a feasible layer assignment for the instance with 4-way junctions. Consider Figure 6b: a 4-way junction is replaced by a series of 3-way junctions. This replacement is performed in an ϵ-region around the former 4-way junction. Let us call the segments by their orientation s_n, s_w, s_s, and s_e in counterclockwise order around the junction. We split each segment and reduce their length by ϵ (two ϵ for s_s). We add new, so-called, ϵ-segments with length ϵ and connect s_n, s_w, s_s, and s_e as shown in Figure 6b. All artificial segments are critical and connected by free edges which correspond to the introduced artificial junctions. As we are allowed to place vias at junctions we prefer the placement of vias to those 4-way junctions. By appropriately adjusting the edge weights in the associated conflict graph, we prefer the placement on these junctions. The artificial junctions will vanish in the postprocessing and vias will possibly be placed on the original junction. This replacement allows for a junction degree up to eight (if the layout is restricted to grid based layouts). Moreover, at the endpoints of each ϵ-segment the junction degree is at most three. This replacement can be done in linear time in a preprocessing step. In a postprocessing step we must shrink these ϵ-replacements back to the former 4-way junction which can be done in linear time as well.

3.4 Pin Layer Preferences

Another constraint comes into play since some pins might have pre-described layer assignments. This means that we know beforehand on which layer a pin must be placed in a feasible two-layer assignment. In theory, one can introduce *pin criticals* at pins. Further one introduces a supervertex s in the reduced conflict graph and connects the corresponding (pin) vertices to it. Supervertex s is assigned to one preferred layer. Edges $(s, v), v \in V \setminus \{s\}$ get a very large weight if the corresponding pin critical has to be assigned to a different layer, or a very small weight in case the corresponding pin critical has to be on the same layer. Apart from very special cases, the reduced conflict graph then is not planar anymore, but almost planar. Barahona [2] has shown that the max-cut problem is already NP-hard on almost planar graphs. Nevertheless, one may follow the construction above. In order to be able to apply the basic algorithm introduced above, one has to find a maximum planar subgraph that includes the reduced conflict graph and as many edges as possible of the form (s, v). The maximum cut on this graph yields a two-layer assignment in which some of the pins with layer preferences are placed on their preferred layer. It might be necessary to place additional vias in order to meet all preferences. However, it turns out experimentally that this approach usually leads to solutions far away from the optimum.

Here, we propose a different strategy for handling pin preassignments within the basic scheme. We do not introduce any additional criticals but place vias at pins if they are not placed on their preferred layer in a postprocessing step. Although this

is a straightforward approach, the computational results show that this strategy is superior to the subgraph one.

3.5 Layer Preferences (Blockages)

Sometimes it is necessary that some segments have to be placed on one specific layer. This situation occurs for example if there is a *blockage* on the other layer. We say such a segment has a layer preference. Via minimization and layer preferences are conflicting goals. By allowing more vias, more segments may be placed on their preferred layer. On the other hand, minimizing the number of vias may yield many segments placed on the wrong layer which is infeasible in practice. We extend the via minimization scheme by another sweep line phase. In this sweep line we find all intersections between blockages and segments. Thus, we gain two pieces of information: *block criticals*, that are critical intervals where no via can be placed due to blockages, and a preference to one layer. Again we could proceed as discussed in Section 3.4 in the case with pin preassignment. But again this may yield solutions far away from the optimum. Instead we only introduce block criticals and solve the via minimization problem with them. In a postprocessing step we identify those criticals that were placed on the wrong layer and repair the solution by introducing new vias. As we see in the computational results in Section 4, this approach works well in practice.

4 Computational Results

In this section we report our computational results on real-world instances. The chips used are current ASIC designs and were kindly provided by our cooperation partners at IBM®. The initial wiring was constructed by BonnRoute®[14], which is part of the BonnTools framework [18] and is also used by IBM® in the design of these chips. In the implementation, we use the max-cut implementation presented in [19] as base routine, together with the modifications explained in the above sections. We compare the via numbers with those of the initial BonnRoute®wiring. Further, we used the LEDA library [20] for the intersection test of segments.

The chips used for the instances are from current designs at the gridless 65nm technology node. They have different size and a different routing layout. Most importantly, we also include the information about pin preferences, design rules and blockages. In each case, we used the metal2 and metal3 wiring layers for the experiments. Higher layers are usually used for longer distances and thus provide less room for optimization. As metal1 is usually reserved for circuit internal structures, it does not contain many wire segments. The generated results are feasible and also consider the practical constraints. Our experiments were performed on Intel® Xeon© CPU X5680 3.33GHz (48GB RAM) (running under Debian Linux 5.0).

In Table 1 we report detailed information and results for the different representative chips. The transient routing inputs have between 13,747 and 64,098 many nets with varying number of segments (60,427 up to 661,541) and blockages (52,336 up to 2,146,275). We also give the number of occurring 4-way junctions and the number

Table 1: Results for via minimization on real-world chips. Pin preassignment is taken into account in a postprocessing step. Blockade information are processed from the beginning. The column with label *CC* report the number of connected components in the reduced conflict graph. Entries with negative percentage indicate the percentage of vias that our solution needs additionally, when compared to the currently used solution. Finally, we report the size of the largest component in brackets.

				4-		reduced conflict graph					via	time
chip	nets	segments	blockages	ways	vias	$\|V\|$	($\|V_{\text{largest}}\|$)	$\|E\|$	($\|E_{\text{largest}}\|$)	CC	red. [%]	[secs]
Gertrud1	60,268	316,692	2,146,275	462	97,196	137,113	(103,239)	185,044	(165,710)	17,544	-19.80	107.28
Gertrud2	58,295	287,584	2,146,275	320	86,709	120,933	(84,959)	155,670	(135,498)	18,823	-20.28	99.09
Inaya	64,098	661,541	1,449,723	694	233,803	307,174	(290,516)	483,398	(475,073)	9,417	-34.57	171.34
Joe	13,747	60,427	52,336	0	26,594	22,269	(21,300)	39,123	(38,763)	687	28.89	3.87
Joe2	15,079	79,618	52,336	126	26,507	32,373	(29,643)	49,441	(47,877)	1,455	9.06	6.02
Lucius1	51,953	327,579	717,953	182	118,391	260,196	(253,359)	431,027	(426,295)	3,066	33.49	59.98
Lucius2	56,868	408,537	932,384	469	145,938	183,373	(171,668)	295,814	(290,291)	6,984	-25.88	81.00
Renate	42,624	350,106	933,397	484	120,995	212,602	(200,602)	326,194	(318,874)	5,877	25.63	71.43

of vias used in the original transient layout. Finally, we report the size of the reduced conflict graph. Clearly, the reduced conflict graphs are sparse. Furthermore, they consist of several connected components that are processed sequentially. We observe that there is always one big component of size of about 90% of the reduced conflict graph. Moreover, there are many small components with up to three vertices. The latter make up between 91% and 96% of all connected components. Applying the max-cut algorithm can be done fast which can be seen in the small total running-times. On four out of eight chips we can drastically reduce the number of vias needed. On chip Joe2 about 9% less vias are needed, on Renate we could save about one fourth of the vias currently used. Similarly, on chips Lucius1 and Joe, a reduction of about 30% was possible. On the other hand, if we could not achieve a reduction of the number of vias, we needed between 20% (Gertrud1) and 35% (Inaya) more vias. This can be explained by the high number of intersections of segments with blockages on the corresponding chips. For the chips Gertrud1 and Gertrud2 there are more than $8.2 * 10^6$ of them, on Inaya more than $2.5 * 10^6$, and on Lucius1 still more than $1.3 * 10^6$. Due to the large number of blockages on those chips, it is preferable to place horizontal segments to one layer and vertical segments to the other.

In order to evaluate different ways of modeling pin preassignments, we also report in Table 1 results when pin preassignment is done within a postprocessing step. We did not use any other heuristic method for the pin preassignment constraint as proposed in Section 3.4. However, to compare this approach to the one proposed in Section 3.4 we present in Tables 2 - 4 results obtained by introducing pin criticals. In Table 2 we introduced them for all pins, while in Table 3 we only introduced them for pins on layer 1, and in Table 4 only for pins on layer 2 pin criticals were used.

Table 2: Results for via minimization. Pin preassignment for both layers is taken into account by a heuristic approach as explained in Section 3.4. Blockage information are processed from the beginning. Positive (negative, resp.) numbers indicate the percentage of vias saved (additionally used, resp.), when compared to the current solution. The column with label *CC* report the number of connected components in the reduced conflict graph. Additionally to the number of connected components, we report the size of the largest component in brackets.

	reduced conflict graph					via	time
chip	$\|V\|$	($\|V_{\text{largest}}\|$)	$\|E\|$	($\|E_{\text{largest}}\|$)	CC	red. [%]	[secs]
Gertrud1	242,618	(166,501)	279,697	(228,972)	28,396	-78.79	215.71
Gertrud2	223,313	(141,566)	246,583	(192,105)	30,290	-85.99	196.80
Inaya	446,597	(409,068)	617,702	(593,625)	14,536	-55.88	286.54
Joe	45,042	(40,473)	60,762	(57,936)	1,821	-11.00	5.27
Joe2	55,708	(47,573)	71,068	(65,807)	3,163	-22.69	8.47
Lucius1	366,147	(336,379)	527,820	(509,315)	12,224	-3.18	161.26
Lucius2	275,691	(247,115)	383,847	(365,738)	11,269	-46.84	143.28
Renate	316,373	(279,339)	421,478	(397,529)	14,364	-17.01	150.37

Introducing pin criticals for all pins yields worst performance. As we try to find a planar subgraph with many vertices corresponding to pin criticals we obviously miss some of those. Hence, the solution on the chosen planar subgraph might be far away from any optimum solution. This is worse when using pin criticals for all pins. This approach thus does not reduce the number of vias needed on any chip. The solutions

Table 3: Results for via minimization with pin preassignment only on layer 1. Pin preassignment is taken into account by a heuristic approach as explained in Section 3.4. Blockage information are processed from the beginning. Positive (negative, resp.) numbers indicate the percentage of vias saved (additionally used, resp.), when compared to the current solution. The column with label *CC* report the number of connected components in the reduced conflict graph. Additionally to the number of connected components, we report the size of the largest component in brackets.

	reduced conflict graph					via	time
chip	$\|V\|$	($\|V_{\text{largest}}\|$)	$\|E\|$	($\|E_{\text{largest}}\|$)	CC	red. [%]	[secs]
Gertrud1	149,879	(110,337)	194,077	(172,808)	21,277	-23.15	116.11
Gertrud2	133,036	(91,455)	164,109	(141,994)	22,487	-23.61	110.73
Inaya	327,339	(305,193)	499,922	(489,750)	13,058	-39.47	184.62
Joe	23,609	(22,128)	40,056	(39,591)	1,094	27.31	194.22
Joe2	34,724	(30,956)	51,023	(49,190)	2,224	8.36	6.22
Lucius1	287,358	(272,484)	451,657	(445,420)	9,598	30.61	84.84
Lucius2	200,707	(183,710)	309,851	(302,333)	10,281	-29.25	103.77
Renate	221,091	(206,320)	332,358	(324,510)	8,202	24.82	76.34

Table 4: Results for via minimization with pin preassignment only on layer 2. Pin preassignment is taken into account by a heuristic approach as explained in Section 3.4. Blockage information are processed from the beginning. Positive (negative, resp.) numbers indicate the percentage of vias saved (additionally used, resp.), when compared to the current solution. The column with label *CC* report the number of connected components in the reduced conflict graph. Additionally to the number of connected components, we report the size of the largest component in brackets.

	reduced conflict graph					via	time
chip	$\|V\|$	($\|V_{\text{largest}}\|$)	$\|E\|$	($\|E_{\text{largest}}\|$)	CC	red. [%]	[secs]
Gertrud1	229,852	(159,403)	267,171	(221,874)	28,156	-76.07	238.35
Gertrud2	211,210	(135,070)	234,715	(185,609)	30,055	-82.91	192.47
Inaya	426,432	(394,391)	597,620	(578,948)	14,453	-51.73	281.39
Joe	43,702	(39,645)	59,489	(57,108)	1,754	-9.63	5.61
Joe2	53,357	(46,260)	68,790	(64,494)	3,090	-28.77	10.48
Lucius1	338,985	(317,254)	500,951	(490,190)	11,931	0.65	123.38
Lucius2	258,357	(235,073)	366,696	(353,696)	11,086	-43.14	139.60
Renate	307,884	(273,703)	412,999	(391,893)	14,354	-6.87	159.44

needed up to twice as many vias as in the original transient routing. Moreover, the running-times are considerably larger. The reduced conflict graphs are larger but there are more subgraphs.

Using only pin criticals for pins on layer 1 shows better results, see Table 3. Nevertheless, they are worse than without any pin criticals, compare Table 1. The running-times are comparable as the sizes of the reduced conflict graphs only differ slightly. The picture is different when considering pin criticals only on layer 2. The results are similar to those obtained if pin criticals are used for all pins. Nevertheless, on chip Lucius1 a marginally reduction of the number of vias needed can be observed, in return the running-time was twice as long as without pin criticals.

As a summary, applying the presented method can be done fast (less than three minutes) on real-world chip instances. Best results are found when pin preassignment for both layers is taken into account by a heuristic approach as explained in Section 3.4. The corresponding computational results are presented in Table 1. Then, either we obtain a better layer assignment with considerable less vias or we verified that the given transient routing is good in terms of the number of used vias.

5 Conclusion

We described a combinatorial approach for via minimization on two layers in VLSI chip design. The method was originally proposed in [8, 23, 3, 15]. Here, we presented an implementation which take into account practical constraints like pin preassignment and blockages. Further, we also incorporated a natural way to deal with 4-way junctions, and replaced those by a chain of 3-way junctions. Using a fast maximum cut algorithm for planar graphs [19] as exact core routine, the via minimization approach achieves good results on real-world chips. Solutions can be determined within a few minutes of CPU time only. The solution either yields a considerable reduction in the number of vias, or the instance possesses a very dense packing of blockages. We showed that there is space for optimization in large real-world chips. Indeed, in VLSI routing, the number of vias used for wiring interconnects is an important figure when estimating the quality of a result, especially in terms of yield. With the method outlined above, the number of vias in modern real-world chips can be kept low.

References

1. Alpert, C., Li, Z., Moffit, M., Nam, G., Roy, J., Tellez, G.: What makes a design difficult to route. In: Proc. ISPD '10 (2010)
2. Barahona, F.: On the Computational Complexity of Ising Spin Glass Models. Journal of Physics A: Mathematical and General **15**(10), 3241–3253 (1982)
3. Barahona, F., Grötschel, M., Jünger, M., Reinelt, G.: An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design. Operations Research **36**(3), 493–513 (1988)
4. Bentley, J.L., Ottmann, T.A.: Algorithms for Reporting and Counting Geometric Intersections. IEEE Transaction on Computers **28**, 643–647 (1979). DOI http://dx.doi.org/10.1109/TC.1979.1675432
5. Chang, C.C., Cong, J.: An Efficient Approach to Multi-layer Layer Assignment with Application to VIA Minimization. Design Automation Conference (1997). DOI 10.1109/DAC.1997.597216. 4 pages
6. Chang, K.C., Du, D.H.C.: Efficient Algorithms for Layer Assignment Problem. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **6**, 67–78 (1987). DOI 10.1109/TCAD.1987.1270247
7. Chang, K.C., Du, D.H.C.: Layer Assignment Problem for Three-Layer Routing. IEEE Transactions on Computers **37**, 625–632 (1988)
8. Chen, R.W., Kajitani, Y., Chan, S.P.: A Graph-Theoretic VIA Minimization Algorithm for Two-Layer Printed Circuit Boards. IEEE Transactions on Circuits and Systems **30**, 284–299 (1983). DOI 10.1109/TCS.1983.1085357
9. Choi, H.A., Nakajima, K., Rim, C.S.: Graph bipartization and via minimization. SIAM Journal on Discrete Mathematics **2**, 38–47 (1989). DOI 10.1137/0402004
10. Chou, Y.C., Lin, Y.L.: A Graph-Partitioning-Based Approach for Multi-Layer Constrained VIA Minimization. International Conference on Computer-Aided Design pp. 426–429 (1998). DOI 10.1109/ICCAD.1998.742908
11. Ciesielski, M.J., Kinnen, E.: An Optimum Layer Assignment for Routing in ICs and PCBs. In: Proceedings of the 18th Design Automation Conference, DAC '81, pp. 733–737 (1981)
12. Fouilhoux, P., Mahjoub, A.R.: An Exact Model for Multi-Layer Constrained VIA Minimization. Tech. rep. (2006)
13. Fouilhoux, P., Mahjoub, A.R.: Solving VLSI Design and DNA Sequencing Problems Using Bipartization of Graphs. Computational Optimization and Applications pp. 1–33 (2010). DOI 10.1007/s10589-010-9355-1
14. Gester, M., Müller, D., Nieberg, T., Panten, C., Schulte, C., Vygen, J.: BonnRoute: Algorithms and data structures for fast and good VLSI routing. preprint, University of Bonn (2011)
15. Grötschel, M., Jünger, M., Reinelt, G.: VIA Minimization with Pin Preassignments and Layer Preference. Zeitschrift für Angewandte Mathematik und Mechanik **69**(11), 393–399 (1989)

16. Hashimoto, A., Stevens, J.: Wire Routing by Optimizing Channel Assignment Within Large Apertures. In: Proceedings of the 8th Design Automation Workshop, DAC '71, pp. 155–169 (1971). DOI http://doi.acm.org/10.1145/800158.805069
17. Kajitani, Y.: On VIA Hole Minimization of Routing on a 2-Layer Board. In: Proceedings of the IEEE International Conference on Circuits and Computers, ICCC '80, pp. 295–298 (1980)
18. Korte, B., Rautenbach, D., Vygen, J.: BonnTools: Mathematical Innovation for Layout and Timing Closure of Systems on a Chip. In: Proceedings of the IEEE, vol. 95, pp. 555–572 (2007). DOI 10.1109/JPROC.2006.889373
19. Liers, F., Pardella, G.: Partitioning Planar Graphs: a Fast Combinatorial Approach for Max-Cut. Computational Optimization and Applications pp. 1–22 (2010). DOI:10.1007/s10589-010-9335-5
20. Mehlhorn, K., Näher, S.: LEDA: A Platform for Combinatorial and Geometric Computing. Cambridge University Press (1999)
21. Naclerio, N.J., Masuda, S., Nakajima, K.: Via minimization for gridless layouts. In: Proceedings of the 24th ACM/IEEE Design Automation Conference, DAC '87, pp. 159–165 (1987). DOI http://doi.acm.org/10.1145/37888.37912
22. Naclerio, N.J., Masuda, S., Nakajima, K.: The VIA Minimization Problem is NP-Complete. IEEE Transactions on Computers **38**, 1604–1608 (1989). DOI http://dx.doi.org/10.1109/12.42135
23. Pinter, R.Y.: Optimal Layer Assignment for Interconnect. Advances in VLSI and Computer Systems **1**, 123–137 (1984)
24. Shamos, M.I., Hoey, D.: Geometric Intersection Problems. In: 17th Annual Symposium on Foundations of Computer Science, FOCS, pp. 208–215. IEEE (1976)