Stitch-Aware Routing for Multiple E-Beam Lithography*

Shao-Yun Fang[1], Iou-Jen Liu, and Yao-Wen Chang[1,2]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan
{yuko703, yrliu}@eda.ee.ntu.edu.tw; ywchang@cc.ee.ntu.edu.tw

ABSTRACT

Multiple e-beam lithography (MEBL) is one of the most promising next generation lithography (NGL) technologies for high volume manufacturing, which improves the most critical issue of conventional single e-beam lithography, throughput, by simultaneously using thousands or millions of e-beams. For parallel writing in MEBL, a layout is split into stripes and patterns are cut by stripe boundaries, which are defined as *stitching lines*. Critical patterns cut by stitching lines could suffer from severe pattern distortion or even yield loss. Therefore, considering the positions of stitching lines and avoiding stitching line-induced bad patterns are required during layout design. In this paper, we propose the first work of stitch-aware routing framework for MEBL based on a two-pass bottom-up multilevel router. We first identify three types of stitching line-induced bad patterns which should not exist in an MEBL-friendly routing solution. Then, stitch-aware routing algorithms are respectively developed for global routing, layer/track assignment and detailed routing. Experimental results show that our stitch-aware routing framework can effectively reduce stitching line-induced bad patterns and thus may not only improve the manufacturability but also facilitate the development of MEBL.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids

General Terms

Algorithms, Design, Performance

Keywords

Multiple electron beam lithography, stitch, routing, manufacturability

1. INTRODUCTION

E-beam lithography (EBL) is one of the most expected Next Generation Lithography (NGL) technologies for overcoming the manufacturing limitations of conventional optical lithography. However, the relatively low throughput due to the maskless direct write process constrains EBL from high volume manufacturing. Thus, EBL was only applied to few applications such as photomask fabrication [15]. In recent years, the concept of multiple e-beam lithography (MEBL) has been proposed, which utilizes massively parallel exposure with thousands or even millions of beams to dramatically improve the throughput. Also, several innovative MEBL systems have been under development and have shown very promising lithography performance and cost effectiveness [11, 13, 14, 17].

*This work was partially supported by IBM, SpringSoft, TSMC, Academia Sinica, and NSC of Taiwan under Grant No's. 101-2221-E-002-191-MY3, NSC100-2221-E-002-088-MY3, NSC 99-2221-E-002-207-MY3, and NSC 99-2221-E-002-210-MY3.

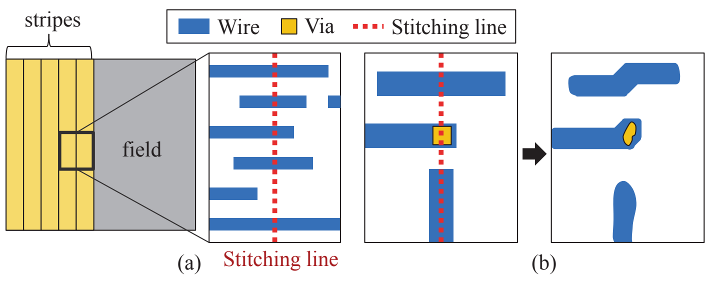

Figure 1: Layout division and overlay error in MEBL. (a) A layout is split into stripes and the stripe boundaries are defined as the stitching lines. (b) Features cut by stitching lines suffer from different degrees of pattern distortion.

Due to the deflection limitation of each beam and parallel writing strategies in MEBL, a layout (a main field) is split into stripes (subfields) as shown in Figure 1(a), and we define the stripe boundaries as the *stitching lines*. Since patterns in different stripes are written by different beams or in different writing passes, a pattern cut by a stitching line suffers from overlay error between two beams or two writing passes [7, 16]. Note that the overlay error could cause different impacts on different types of patterns cut by stitching lines. As illustrated in Figure 1(b), a horizontal wire can be patterned well even if an overlay error exists. On the other hand, some patterns with critical dimension, such as vias or vertical wires, can have severe pattern distortion and electrical variation due to the overlay error. Therefore, it is desirable to consider stitching lines for MEBL-friendly layout designs to enhance manufacturability. However, to the best of our knowledge, no previous work has addressed the stitching line-induced printability problems during physical design for MEBL.

In current semiconductor manufactruing, metal layers become one of the most critical parts with respect to reliability, manufacturability, and circuit performance, and thus routing plays a crucial role in the VLSI design flow. In MEBL, routing without considering stitching lines may cause stitching line-induced bad patterns. As shown in Figure 2(a), without stitching line consideration, a via is cut by the stitching line on the wire A, and a part of the wire B is vertically routed on the stitching line. Another undesired pattern occurs on the wire C, which is a short wire segment cut by the stitching line with a landing via. We define this type of patterns as *short polygons*. Short polygons may also cause severe manufacturing defects, which will be explained in Section 2. Figure 2(b) shows a better routing result, where no stitching line-induced bad pattern is produced. Avoiding vias cut by stitching lines and avoiding wires vertically routed on stitching lines are not difficult. For example, removing routing tracks vertically overlapped with stitching lines can prevent wires from vertically routing on stitching lines. However, avoiding the generation of short polygons is not trivial. In fact, considering this type of bad patterns could significantly increase the design complexity.

In this paper, we propose the first work of stitch-aware routing for MEBL, which minimizes the number of stitching line-induced bad patterns during routing. The framework is based on a two-

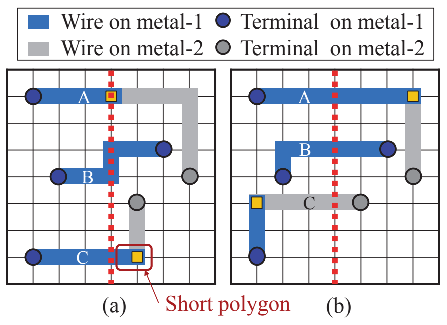

Figure 2: Routing with and w/o stitching consideration. (a) Stitching line-induced bad patterns are generated without considering stitching lines during routing. (b) A better routing solution derived from a stitch-aware router.

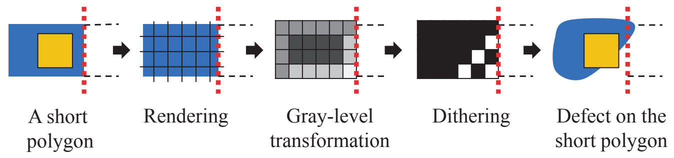

Figure 3: The rasterization process of a short polygon. A severe defect occurs due to dithering with error diffusion.

pass bottom-up multilevel router (similar to [3] for double via optimization). We first identify three types of stitching line-induced bad patterns and establish three corresponding stitch-aware routing constraints. Then, the stitch-aware routing algorithms are respectively proposed in each routing stage: global routing, layer/track assignment and detailed routing. Experimental results show that our algorithms can effectively avoid the generation of stitching line-induced bad patterns for MEBL.

The rest of this paper is organized as follows: Section 2 introduces the three routing constraints. In Section 3, the stitch-aware routing algorithms in global routing, layer/track assignment, and detailed routing are respectively presented. Section 4 reports our experimental results. Finally, we conclude our work in Section 5.

2. STITCH-AWARE ROUTING CONSTRAINTS

As mentioned in Section 1, patterns cut by stitching lines suffer from overlay errors between two different beams or two different writing passes. Although these pattern segmentations are inevitable during circuit design, stitching lines should avoid cutting critical patterns to reduce severe pattern distortion or even yield loss. For example, as mentioned in Section 1, vias should not be cut by stitching lines and wires should not vertically route on stitching lines.

Another type of stitching line-induced bad patterns, *short polygons*, is due to the data preparation flow in MEBL. Because of the maskless lithography process, *rasterization* is required to transform a layout into a pixel-based black/white bitmap, and thus patterns can be exposed on a wafer by controlling each independent beam to be "on" or "off" [7, 10]. Rasterization consists of two major steps: (1) *rendering* followed by (2) *dithering* with error diffusion. In rendering, a layout is sliced into grids, and patterns are converted into pixel-based gray-level data with intensity proportional to the pattern coverage in each pixel. Then, in dithering, the resulting gray-level bitmap is transformed into a black/white bitmap. The error of each pixel due to dithering is not neglected but diffused to its neighboring unprocessed pixels.

A short polygon may cause a severe defect after the rasterization process. Figure 3 shows an example. The short polygon cut by the stitching line undergoes rendering and dithering during the data preparation flow. Due to the error diffusion process,

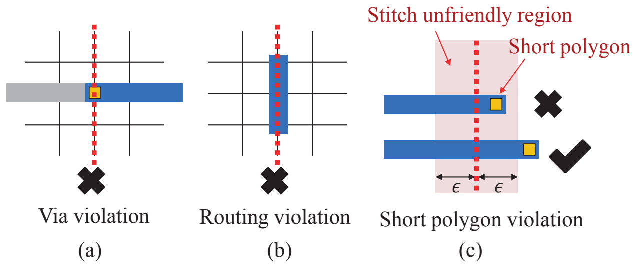

Figure 4: Three routing constraints for stitch-aware routing. (a) Via constraint. (b) Vertically routing constraint. (c) Short polygon constraint.

the short polygon has irregular pixels on the bottom-right corner. These problematic pixels could account for a large percentage of the pixels associated with the short polygon and thus can result in serious pattern distortion after e-beam exposure. Then, the misalignment between the polygon and the via becomes a circuit defect or causes unacceptable electrical variation. Therefore, short polygons with landing vias should be avoided in a routing solution for better MEBL control.

Hence, given a set of stitching lines, we define the following three routing constraints:

- Via constraint: vias cannot be cut by stitching lines (see Figure 4(a)).
- Vertical routing constraint: wires cannot vertically route on stitching lines (see Figure 4(b)).
- Short polygon constraint: vias should not land on short polygons. As illustrated in Figure 4(c), we define the area within the distance ϵ from a stitching line as the *stitch unfriendly region* of the stitching line. A horizontal wire has a short polygon violation if it satisfies the following two conditions: (1) The wire is cut by a stitching line. (2) At least a line end of the wire lies in the corresponding stitch unfriendly region with a landing via. Therefore, in Figure 4(c), the upper wire has a short polygon violation, and the lower wire is a preferred routing instance without any violation.

Note that in our routing framework, the via constraint and the vertical routing constraint are hard constraints that a routing solution must always satisfy, and the short polygon constraint is a soft constraint that our routing framework should optimize.

3. STITCH-AWARE ROUTING FRAMEWORK

In this section, stitch-aware global routing, layer/track assignment, and detailed routing are presented in the following subsections.

3.1 Stitch-Aware Global Routing

In the global routing stage, a routing plane is first divided into global tiles and transformed into a routing graph, in which a vertex represents a global tile and each pair of adjacent global tiles is connected by an edge, as shown in Figure 5(a). Then, nets sequentially find their global routing paths on the graph with minimized routing costs. The routing cost of a routing path is usually computed according to the routing congestion on the path.

Resource estimation in global routing for MEBL is quite different from conventional routing problems due to the existence of stitching lines. For example, in Figure 5(b), the capacity of each boundary (the maximum number of wires that can pass through the boundary) of the global tile is originally six without considering stitching lines. However, the capacities of the top boundary and the bottom boundary are reduced by one since no wire can route on the track occupied by the stitching line due to the vertical routing constraint. Furthermore, it is undesirable that many line ends of vertical segments lie in the same tile. As shown in Figure 5(b), only two vertical tracks are not in stitch unfriendly regions. If there are three vertical segments whose line ends lie in the tile, at least one line end will lie in the stitch unfriendly region, and the line end may cause a short polygon violation on the connected horizontal wire, as the segment C in Figure 5(b).

To consider both of the situations, in a global routing graph, each edge is assigned an edge capacity indicating the maximum number of wires that can pass through the tile boundary without overflow, and each vertex is also assigned a vertex capacity denoting the number of tracks not in stitch unfriendly regions. Then, the cost of an edge e_i ($\psi_e(i)$) and the cost of a vertex v_j ($\psi_v(j)$) are respectively defined as follows:

$$\psi_e(i) = 2^{d_e(i)/c_e(i)} - 1, \quad (1)$$

$$\psi_v(j) = 2^{d_v(j)/c_v(j)} - 1, \quad (2)$$

where $c_e(i)$ is the capacity of e_i, $c_v(j)$ is the capacity of v_j, $d_e(i)$ is the demand of e_i, which is the number of segments that have routed on e_i, and $d_v(j)$ is the demand of v_j, which is the number of line ends that have lain on v_j. Thus, the routing cost of a global routing path is the summation of the vertex costs and the edge costs in the path.

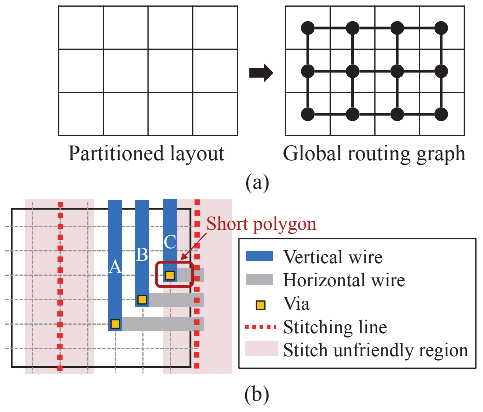

Figure 5: Global routing model and routing resource estimation for MEBL. (a) A layout is divided into global tiles and transformed into a graph model. (b) The routing resource is reduced due to the stitching lines.

3.2 Stitch-Aware Layer Assignment

Layer/track assignment has been proven as an effective intermediate stage between global routing and detailed routing [1, 6] for improving the routing quality of high complexity designs. In addition, many manufacturability issues can be optimized during layer/track assignment, such as crosstalk, antenna effect, and wire density uniformity [4, 9, 12, 18]. In our work, stitch-aware layer and track assignment algorithms are also proposed for optimizing stitching line-induced bad patterns.

In layer assignment, we assign the vertical (horizontal) segments in a column (row) panel to different vertical (horizontal) routing layers. A column (row) panel is defined as a column (row) of global tiles in a global routing graph. The conventional objective in layer assignment is to uniformly distribute segments in a panel. However, as mentioned in Section 3.1, line ends of segments should be also scattered to different layers to avoid generating short polygons.

To solve the stitch-aware layer assignment problem, we first construct a segment conflict graph for each panel, in which a vertex v_i represents a segment i and an edge connecting two vertices if the two segments intersect in some tiles. For a column panel, we set an edge weight $w(v_i, v_j)$ for each edge (v_i, v_j) as follows:

$$w(v_i, v_j) = D_{segment}(v_i, v_j) + D_{end}(v_i, v_j), \quad (3)$$

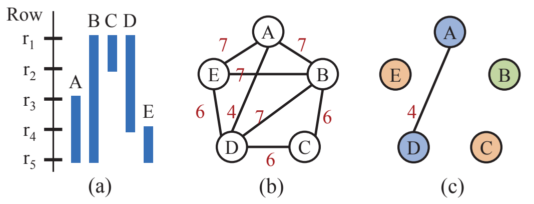

Figure 6: Layer assignment considering segment and line-end uniformities. (a) A set of segments in a vertical panel. (b) The corresponding segment conflict graph. (c) A layer assignment solution by solving the maximum-cut k-coloring problem.

where $D_{segment}(v_i, v_j)$ is the maximum segment density in the rows where the segment i and the segment j are overlapped, and $D_{end}(v_i, v_j)$ is the maximum line-end density in the rows where the line ends of i and j are overlapped. (Note that we simply remove the second item in Equation (3) for row panels since we consider line-end densities only in column panels.) Figure 6(b) shows a conflict graph for the segments in a column panel shown in Figure 6(a). To uniformly distribute segments and line ends to k layers, the layer assignment problem can be solved by finding a maximum-cut k-coloring solution [5] of the segment conflict graph, which is equivalent to finding a k-coloring solution with the minimum total edge weight [5]. Figure 6(c) shows a three-coloring solution with the minimum total edge weight of the segment conflict graph.

Since the maximum-cut k-coloring problem is NP-complete [5], previous work has proposed a heuristic approach that first constructs a maximum spanning tree on a conflict graph and then solves the k-coloring problem on the tree. Note that a tree is always k-colorable when $k \geq 2$. This heuristic can solve the maximum-cut k-coloring problem well as k equals two; however, as k is greater than two, solving the maximum-cut k-coloring problem with the maximum spanning tree approach may degrade the solution quality since more edges can be simultaneously considered as more colors are available. As illustrated in Figures 7(a) and (b), if three vertical layers are available, after constructing a maximum spanning tree and three-coloring the tree according to the tree level of each vertex, a layer assignment solution is generated with total edge weight equal to 13.

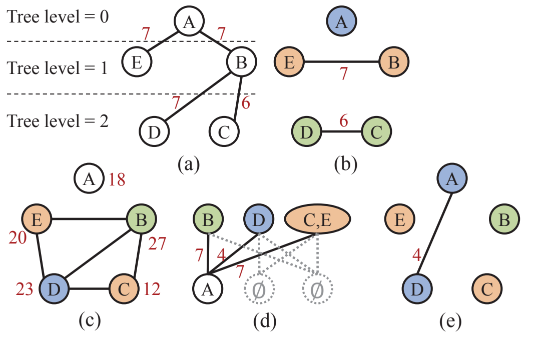

Figure 7: Heuristics for solving the maximum-cut k-coloring problem. (a)(b) The maximum spanning tree approach. (c)(d)(e) Our algorithm that can generate a better solution.

In this work, we propose another heuristic algorithm to get better solutions. We first compute the vertex weight for each vertex by summing the weights of the incident edges. Then, we find a set of k-colorable vertices with the maximum total vertex weight. Although this problem is NP-complete in general graphs, it can be solved in polynomial time for segment conflict graphs, which are interval graphs, by using a minimum cost flow algorithm [2]. As shown in Figure 7(c), $V_1 = \{v_B, v_C, v_D, v_E\}$ is a

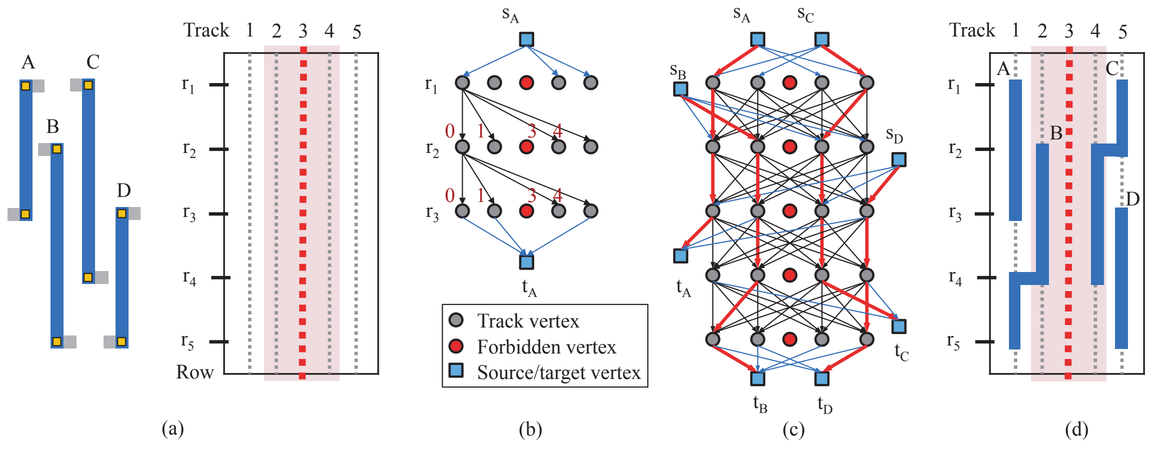

Figure 8: The ILP-based track assignment approach. (a) A track assignment instance. (b) The multi-commodity flow model of the segment A. (c) The multi-commodity flow model of all segments and the solution derived from the ILP formulation. (d) The corresponding track assignment solution.

three-colorable vertex set in the segment conflict graph with the maximum total vertex weight, and $\{v_B\}, \{v_D\}$ and $\{v_C, v_E\}$ are the three-coloring groups of V_1. The algorithm then finds the next k-colorable vertex set with the maximum total vertex weight on the remaining graph. To merge the coloring groups of the two vertex sets, a perfect bipartite matching algorithm is applied to minimize the total conflict edge weight. As illustrated in Figure 7(d), two pseudo coloring groups $\emptyset$ are first created since only the vertex v_A remains, and thus the three-coloring groups of the second vertex set V_2 are $\{v_A\}, \emptyset$ and $\emptyset$. To combine the coloring groups of V_1 and V_2, a complete bipartite graph is constructed and the edge weights are set as the total conflict edge weight between two groups. By solving the minimum weight perfect bipartite matching problem, coloring groups are merged with the minimum conflict edge weight. The above process is performed iteratively until no vertex is left. Figure 7(e) shows the layer assignment result with smaller total edge weight equal to 4.

3.3 Short Polygon-Avoid Track Assignment

In track assignment, segments of the same layer in a panel are assigned exact track numbers, which is a crucial stage for short polygon avoidance. A desired track assignment solution which can avoid short polygon generation is a track assignment without *bad ends*. A bad end is a line end of a vertical wire segment lying in the stitch unfriendly region of a stitching line, and the connected horizontal wire is cut by the stitching line. For example, the lower end of the wire segment C in Figure 5(b) is a bad end. To derive a track assignment solution without bad ends, an ILP-based algorithm and a graph-based algorithm are proposed, which are detailed in the following subsections. Note that the short polygon-avoiding track assignment algorithms are only applied to column panels. Segments in row panels can be assigned by using conventional track assignment algorithms.

3.3.1 ILP-Based Approach

First, the short polygon-avoiding track assignment problem can be intuitively transformed into a multi-commodity flow model, which is a directed graph $G = (V, E)$. Figure 8(a) shows a track assignment instance. To find an exact track number for the segment A, for example, the multi-commodity flow graph model is constructed as shown in Figure 8(b), where a track vertex represents a track in a global tile, a forbidden vertex is a track occupied by a stitching line, and the source vertex s_A and the target vertex t_A are the top end and the bottom end of the segment A. The source edges connect s_A to the track vertices of the tile where the top end of A lies. Similarly, the target edges connect the track vertices of the tile where the bottom end of A lies to t_A. A source/target edge is removed if the line end becomes a bad end on the corresponding track. For example, s_A causes a bad end if it starts on the second track, and thus the second source edge is removed from the graph. Also, track vertices of adjacent tiles are connected with track edges, and the edge weight of a track edge is set to be the difference of the track numbers of the two track vertices to minimize wirelength and the number of routing bends. The whole multi-commodity flow graph model of the four segments is shown in Figure 8(c). Then, we find a track assignment solution with an ILP formulation. The notation used in our ILP formulation is listed as follows:

- K: a set of segments in a track assignment problem.
- $s(k)$: the source vertex of the segment k.
- $t(k)$: the target vertex of the segment k.
- V_{track} a set of track vertices.
- $w(u, v)$: the weight of the directed edge (u, v).
- $f_k(u, v)$: 0-1 integer variable that denotes if the segment k is routed through the directed edge (u, v).
- $\mathcal{C}$: a set of crossed edge pairs.

Based on the notations, the short polygon-avoiding layer assignment problem can be formulated as follows:

$$\textit{minimize} \quad \sum_{(u,v)\in E} \left(w(u,v) \times \sum_{k\in K} f_k\,(u,v) \right)$$

$$\textit{subject to} \quad \sum_{(s(k),v)\in E} f_k(s(k), v) = 1, \forall k \in K, \quad (4)$$

$$\sum_{(u,t(k))\in E} f_k(u, t(k)) = 1, \forall k \in K, \quad (5)$$

$$\sum_{u\in V} f_k(u, v) = \sum_{w\in V} f_k(v, w), \forall k \in K, \forall v \in V_{track}, \quad (6)$$

$$\sum_{u\in V} \sum_{k\in K} f_k(u, v) \le 1, \forall v \in V_{track}, \quad (7)$$

$$\sum_{k\in K} f_k(u_1, v_1) + \sum_{k\in K} f_k(u_2, v_2) \le 1, \forall\, ((u_1, v_1), (u_2, v_2)) \in \mathcal{C}. \quad (8)$$

The objective of the ILP formulation is to minimize the total edge weight of a flow solution such that the wirelength and

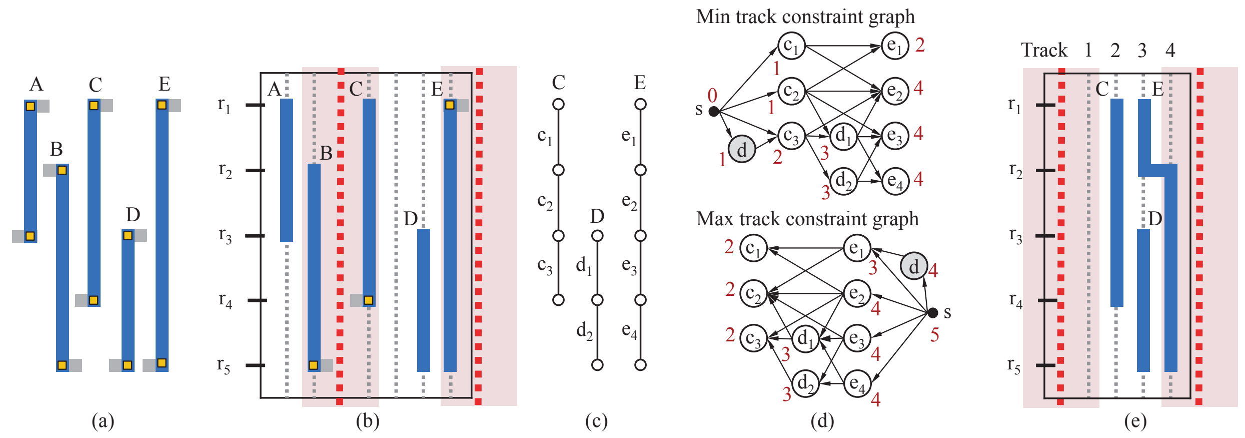

Figure 9: The graph-based track assignment approach. (a) A track assignment instance. (b) A segment order is first determined. (c) The segments C, D and E between two stitching lines are simultaneously considered and are divided into intervals. (d) The feasible track assignment solution space of each interval is computed by using the minimum and maximum track assignment constraint graphs (c) The final track assignment solution of C, D and E.

the number of wire bends can be minimized. Constraint (4) and Constraint (5) ensure that each segment can find a unique path from its source vertex to the target vertex. Constraint (6) is used to guarantee that the number of paths flowing into a node equals that draining from the node. Constraint (7) guarantees that a track in a tile is occupied by at most one segment. Finally, Constraint (8) prevents segments from crossing with each other.

THEOREM 1. *In the above ILP formulation, the number of variables is $O(T^2R)$ and the number of constraints is $O(TR|K| + T^4R)$, where T is the number of tracks in a column panel and R is the number of rows in the global routing graph*

Note that the complexity is dominated by Constraints (6) and (8).

Using "doglegs" to avoid short polygon generation is one of the advantage of the ILP-based approach. However, since the short polygon-avoiding track assignment process is performed for every panel in all vertical layers, the runtime of iteratively solving the ILP formulation may be prohibitively long as the chip size increases. Therefore, we propose another graph-based track assignment heuristic, which can efficiently utilize doglegs for short polygon avoidance.

3.3.2 *Graph-Based Approach*

The graph-based short polygon-avoiding track assignment algorithm first determines the segment order in a panel, and then tries to resolve bad ends with doglegs by using a graph-based algorithm.

The approach starts from assigning longer segments next to stitching lines. As shown in Figures 9(a) and (b), the segments B, C and E are placed adjacent to stitching lines. Longer segments have larger flexibility to avoid short polygon generation by applying doglegs. Then, some bad ends of those longer segments will be generated if the segments do not change their track numbers. For example, the bottom end of B, the bottom end of C, and the top end of E are currently bad ends. After that, segments not overlapped with the bad ends are assigned next to those longer segments such that the bad ends can be easily resolved with doglegs. Therefore, as illustrated in Figure 9(b), the segment A is assigned next to B and the segment D is assigned next to E. For the remaining segments having less impact on bad ends, the track numbers are arbitrarily assigned.

After determining the segment order, doglegs are used to resolve bad ends. A set of segments between two stitching lines are considered at a time. Each segment is first divided into intervals according to global tiles, as the segments C, D, and E shown in Figure 9(c). Then, two constraint graphs are constructed to record the geometry relationship among these intervals. As illustrated in Figure 9(d), the first one is the *minimum track constraint graph*, where a vertex represents an interval and a directed edge (v_i, v_j) indicates that the two intervals are overlapped in the x-direction and the interval i is left to the interval j. A dummy vertex d is created and connected to a vertex v_i if the interval i should not be assigned to the leftmost track. For example, the interval c_3 has a bad end if it is assigned to the leftmost track between the two stitching lines, and thus a dummy vertex is created and connected to v_{c_3} through the edge (d, v_{c_3}) in the minimum track constraint graph. After creating a source vertex s and connecting s to the vertices of the leftmost intervals and the dummy vertices, a longest path algorithm is applied to compute the minimum track number m of each interval, which indicates the leftmost feasible track number. For the *maximum track constraint graph*, the construction is almost the same except that an edge (v_i, v_j) connects the vertex of the interval i right to the vertex of the interval j and a dummy vertex connected to a vertex v_i if the interval i should not be assigned to the rightmost track. A similar algorithm is applied to compute the maximum track number M of each interval. As shown in Figure 9(d), the two numbers of a vertex in the minimum and maximum track constraint graphs give a feasible solution space $[m, M]$ of track assignment for each corresponding interval.

Finally, we sequentially determine the track numbers from the leftmost segment to the rightmost segment between the two stitching lines according to their feasible solution spaces. The wirelength and the number of bends of each segment are greedily optimized during track assignment. For example, all intervals of the segment C are assigned to the second track for wirelength and bend optimizations, and the final assignment solution is shown in Figure 9(e).

3.4 Stitch-Aware Detailed Routing

The final stage of our routing framework is stitch-aware detailed routing, which finds pin-to-segment and segment-to-segment detailed routes with a conventional A*-search routing algorithm [8]. To satisfy the via and vertical routing constraints, wires passing through stitching lines can only route in the x-direction (perpendicular to stitching lines). For minimizing the number of generated short polygons, we give a larger routing cost if a wire in a stitch unfriendly region routing in the z-direction, and thus a detailed path with the minimum number of line ends lying in stitch unfriendly regions will be found.

Table 1: Experimental results that compare the effectiveness and efficiency among different track assignment algorithms.

Circuit	w/o stitch consideration				ILP-based Approach				Graph-based Approach			
	Rout. (%)	#VV	#SP	CPU (s)	Rout. (%)	#VV	#SP	CPU (s)	Rout. (%)	#VV	#SP	CPU (s)
Struct	99.90	584	3282	2	100.00	597	23	505	100.00	597	15	2
Primary1	99.80	212	1431	1	100.00	215	1	8909	100.00	226	0	1
Primary2	98.63	643	5268	6	100.00	667	161	74780	100.00	688	76	6
S5378	96.32	660	584	1	99.10	643	37	3904	99.23	645	9	1
S9234	98.67	312	452	1	99.90	297	30	2983	99.90	294	20	1
S13207	97.43	45	1212	2	99.87	58	92	12767	100.00	58	62	2
S15850	97.14	43	1524	2	99.75	63	124	14912	99.76	62	90	2
S38417	97.89	24	3499	2	NA	NA	NA	> 100000	99.73	41	182	5
S38584	97.71	2163	4020	10	NA	NA	NA	> 100000	99.32	2037	153	11
Dma	96.00	1748	5242	5	99.40	1764	117	3084	99.49	1792	46	6
Dsp1	97.08	2326	5613	4	99.82	2434	59	4743	99.81	2364	17	7
Dsp2	96.82	2367	5770	4	99.74	2389	77	4133	99.75	2402	16	5
Risc1	96.09	3149	8947	11	99.56	3188	161	9883	99.62	3200	44	11
Risc2	96.64	3199	8801	8	99.67	3233	111	8881	99.66	3248	42	9
Comp.	1.000	1.000	1.000	1.0	1.023	1.070	0.034	4159.5	1.022	1.110	0.022	1.1

4. EXPERIMENTAL RESULTS

Our algorithm was implemented in the C++ programming language on a 2.93 GHz Linux workstation with 48 GB memory. The minimum cost flow problem and the minimum weight bipartite matching problem in layer assignment were solved by adopting the LEDA package [20], and the ILP formulation in track assignment was solved by using the CPLEX12.3 library [19]. In our routing framework, the number of stitching lines was set to be that of global tiles in a row, and the stitching lines were uniformly distributed in a layout. In addition, the tracks adjacent to stitching lines fell into stitch unfriendly regions.

We show the effectiveness of avoiding short polygon generation by applying three different track assignment approaches: (1) track assignment without considering stitching lines, (2) track assignment by solving the ILP formulation, and (3) track assignment by applying the graph-based algorithm. Note that all the three approaches use the same stitch-aware algorithms in other routing stages. Two suits of benchmarks were used, the MCNC benchmarks and the Faraday benchmarks. The experimental results are shown in Table 1, where "Rout." gives the routability, "#VV" reports the number of via violations, "#SP" shows the number of short polygon violations, and "CPU" lists the runtime in second. Due to the fixed pin positions of nets, the three approaches have similar numbers of via violations. On the other hand, the results show that by considering stitching lines, both the ILP-based approach and the graph-based approach slightly improve the routability. It is because not considering stitching lines can cause lots of vertical routing violations. Since the routing constraint is a hard constraint a routing solution must satisfy, the violated wires are ripped-up and will be rerouted in the detailed routing stage, which may cause lots of failed nets. Also, both the approaches can effectively reduce the number of short polygon violations by more than 96%. However, the ILP-based approach is too time-consuming to generate a routing solution in a reasonable runtime, and thus the graph-based approach is more appropriate for large scale routing instances. Note that although the ILP formulation can find a track assignment solution without any short polygon if such a solution exists, the ILP-based approach generates more short polygon violations than the graph-based approach. It is because once the ILP formulation fails to find a solution for a track instance, we directly route the segments in the detailed routing stage, which may cause more short polygons. In contrast, if the graph-based approach fails to find a legal solution, we can simply remove the rightmost segment and keep other segments connected.

5. CONCLUSIONS

In this paper, we have proposed the first work of stitch-aware routing framework for MEBL. We first identify three types of stitching line-induced bad patterns which could cause severe pattern distortion, electrical variation, or even yield loss. Then, we provide solutions to avoid generating these bad patterns during each routing stage. Experimental results show that our algorithms can efficiently and effectively reduce the number of short polygons. To remove the via violations due to the fixed pin positions of nets, it is also desirable to develop stitch-aware algorithms at the placement stage, which is our future work to further improve the manufacturability and facilitate the development of MEBL.

6. REFERENCES

[1] Batterywala et al., "Track assignment: a desirable intermediate step betweeen global routing and detailed routing," *Proc. ICCAD*, pp. 59–66, 2002.

[2] Carlisle and Lioyd, "On the k-coloring of intervals," *DAM*, vol. 59, no. 3, pp. 225–235, 1995.

[3] Chen et al., "Full-chip routing considering double-via insertion," *IEEE TCAD*, vol. 27, no. 5, pp. 844–857, 2008.

[4] Chen et al., "A novel wire-density-driven full-chip routing system for CMP variation control," *IEEE TCAD*, vol. 28, no. 2, pp. 193–206, 2009.

[5] Cho et al., "Fast approximation algorithms on maxcut, k-coloring, and k-color ordering for VLSI applications." *IEEE TC*, vol. 47, no. 11, pp. 1253–1266, 1998.

[6] Cong et al., "DUNE–a mulitilayer gridless routing system," *IEEE TCAD*, vol. 20, no. 5, pp. 633–647, 2011.

[7] Hakkennes et al., "Demonstration of real time pattern correction for high throughput maskless lithography," *Proc. SPIE*, vol. 7970, pp. 79701A, 2011.

[8] Hart et al., "A formal basis for the heuristic determination of minimum cost paths," *IEEE SSC*, vol. 4, no. 2, pp. 100-107, 1968.

[9] Ho et al., "Crosstalk- and performance-driven multilevel full-chip routing," *IEEE TCAD*, vol. 24, no. 6, pp. 869–878, 2005.

[10] Hung et al., "Bottlenecks in data preparation flow for multi-beam direct write," *Proc. SPIE*, vol. 8166, pp. 81662C, 2011.

[11] Klein et al., "PML2: the maskless multibeam solution for the 22nm node and beyond," *Proc. SPIE*, vol. 7271, pp. 72710N, 2009.

[12] Lee and Wang, "Simultaneous antenna avoidance and via optimization in layer assignment of multi-layer global routing," *Proc. ICCAD*, pp. 312–318, 2010.

[13] Lin, "Future of multiple-e-beam direct-write systems," *Proc. SPIE*, vol. 8323, pp. 832302, 2012.

[14] McChord et al., "REBL: design progress toward 16 nm half-pitch maskless projection electron beam lithography," *Proc. SPIE*, vol. 8323, pp. 832311

[15] Rizvi, "Handbook of photomask manufacturing technology," Taylor & Francis, 2005.

[16] Ronse, "E-beam maskless lithography: prosoects and challenges," *Proc. SPIE*, vol. 7637, pp. 76370A, 2010.

[17] Wieland et al., "MAPPER: high throughput maskless lithography," *Proc. SPIE*, vol. 7637, pp. 76370F, 2010.

[18] Wu et al., "Antenna avoidance in layer assignemnt," *IEEE TCAD*, vol. 25, no. 4, pp. 734–738, 2006.

[19] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/

[20] The LEDA package. http://www.algorithmic-solutions.com/leda