Stitch Aware Detailed Placement for Multiple E-Beam Lithography

Yibo Lin[1], Bei Yu[2], Yi Zou[1,3], Zhuo Li[4], Charles J. Alpert[4], and David Z. Pan[1]

[1]ECE Department, University of Texas at Austin, Austin, TX, USA
[2]CSE Department, The Chinese University of Hong Kong, NT, Hong Kong
[3]College of Engineering and Applied Sciences, Nanjing University, Nanjing, China
[4]Cadence Design Systems, Inc., Austin, TX, USA

Abstract

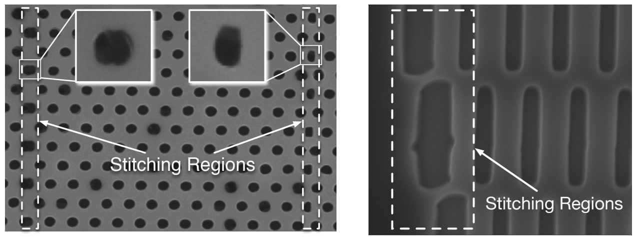
As a promising candidate for next generation lithography, multiple e-beam lithography (MEBL) is able to improve manufacturing throughput using parallel beam printing. In MEBL, a layout is split into stripes and the layout patterns are cut by stripe boundaries, then all the stripes are printed in parallel. If a via pattern or a vertical long wire is overlapping with a stitch, it may suffer from poor printing quality due to the so called *stitch error*; then the circuit performance may be degraded. In this paper, we propose a comprehensive study on the stitch aware detailed placement to simultaneously minimize the stitch error and optimize traditional objectives, e.g., wirelength and density. Experimental results show that our algorithms are very effective on modified ICCAD 2014 benchmarks that zero stitch error is guaranteed while the scaled half-perimeter wirelength is very comparable to a state-of-the-art detailed placer.

I. Introductions

Due to the capability of accurate pattern generation, e-beam lithography (EBL) is a promising candidate for next generation lithography technologies for sub-14nm nodes, along with other techniques such as extreme ultra violet (EUV) and directed self-assembly (DSA) [1]. However, low throughput is still the bottleneck of an EBL system. Recently, an extended EBL technique, multiple e-beam lithography (MEBL), is proposed to improve manufacturing throughput using parallel beam printing [2]. MEBL system utilizes thousands of parallel beams to write multiple layout patterns simultaneously. Industry has already explored different MEBL implementations and has demonstrated promising performance in terms of both lithography accuracy and throughput [3, 4].

In MEBL manufacturing process, a layout is split into stripes, and the boundary between two touching stripes is defined as a stitch line. Each stripe has width of 50∼200 microns, and different stripes are printed simultaneously through different electron beams. Although the parallel writing scheme can dramatically improve the system throughput, it also introduces serious printability issues. That is, each stitch can introduce so called **stitch error**, in an area with width around 15nm [3]. If a pattern is overlapping with a stitch, it may suffer from poor printing quality due to the stitch error. Therefore, if not carefully designed, due to the shape distortion, an MEBL system may confront yield issue or even functional error.

We observe very significant shape distortions on via patterns and long vertical wires. Fig. 1 shows two SEM images of shape distortion on via layer and metal layer, respectively. In Fig. 1(a) we can see that all vias are very regular inside the beam stripes. However, at the stripe boundaries, the vias suffer from obvious distortions and irregular shrinking. In Fig. 1(b) we can see that the vertical wires are malformed in the stitch regions. Similar observations were also reported by Fang et al. [5] that the vertical wires are more susceptible to stitch errors than the horizontal wires.



(a) (b)

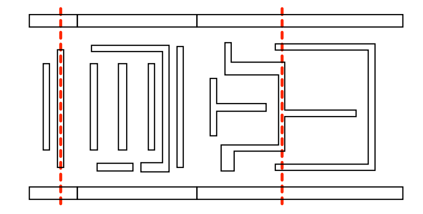
Fig. 1. SEM images of stitch error for (a) via layer and (b) metal layer vertical wires.

There are several methods to minimize the impacts of stitch errors from lithography perspective, e.g., avoiding dividing a critical pattern into adjacent sub-fields [6], using different field sizes [7], or reducing the field size [8]. Recently, Fang et al. [5] considered the stitch error during detailed routing stage. However, detailed routing is a very late stage in physical design flow, thus there may exist some stitch errors difficult to be removed. For instance, stitch errors from vias dropped on pins of a standard cell cannot be optimized during routing stage.

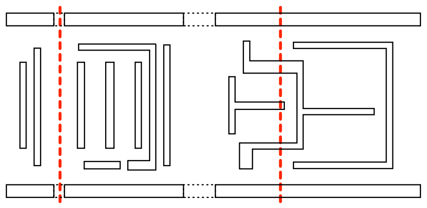
In this work we propose a comprehensive study to consider the stitch error removal in detailed placement. We can directly optimize the positions of both vias and intra-cell vertical wires. In addition, we consider local congestion, thus a router (e.g. [5]) has more routing options to effectively remove stitch errors in higher metal layers. Fig. 2 shows a placement example with three gates, where the density of vertical metal1 segments varies from cell to cell. Some cells are more susceptible to stitch errors as they have vertical wire segments distributed at every site, while other cells have more space to avoid stitch errors. The comparison between Fig. 2(a) and Fig. 2(b) shows that it is possible to smartly avoid stitch errors with small cell movement.

To the best of our knowledge, this is the first work taking stitch errors into consideration in placement stage. Our contributions are summarized as follows.

- We propose a comprehensive detailed placement study to



(a)



(b)

Fig. 2. An example of (a) stitch errors in placement and (b) e-beam friendly placement.

simultaneously minimize the stitch error and optimize traditional objectives, e.g., wirelength and density.

- We develop a swap-based detailed placement engine with an optimal stitch aware single row placement.
- We present an $O(nM)$ pruning technique to speed-up the single row problem, where n and M are number of cells in a row and maximum displacement respectively. Our pruning technique is very generic that it is applicable to conventional placement and other applications.

The rest of this paper is as follows. Section II introduces the stitch constraints and the problem formulation. Section III explains the optimization algorithms in detail. Section IV lists the experimental results, followed by conclusion in Section V.

II. Preliminaries and Problem Formulation

In an MEBL system, stitch lines repeat periodically with equal intervals. If a standard cell is not carefully placed and overlaps with one stitch line, it may suffer from stitch error. In this work we consider three kinds of possible stitch errors, as follows. (1) **Stitch over via**: if a via is cut by a stitch line, it can lead to potential disconnection. (2) **Vertical routing**: a vertical routing segment suffers more from stitch lines than horizontal lines. (3) **Short polygon**: short horizontal routing segment with vias may also result in problem.

To accurately capture a stitch error, we partition each cell into sites with width equal to the poly pitch. Since some intra-cell segments or vias are very susceptible to stitches, we note those sites covered by these segments/vias as *dangerous sites*. For example, Fig. 3 shows the dangerous sites of cell BUF_X8. Note that for simplicity, here only intra-cell segments are illustrated. A stitch error happens iff one dangerous site overlaps with an MEBL system stitch line.

This work adopts scaled half-perimeter wirelength (sHPWL) from ICCAD 2013 placement contest, defined as follows.

$$sHPWL = HPWL \times (1 + \alpha \times P_{ABU}) \qquad (1)$$

where α is set to 1, and HPWL denotes half-perimeter wirelength. P_{ABU} represents *ABU penalty* to evaluate the placement congestion. Please refer to [9] for more details regarding the P_{ABU} calculation.

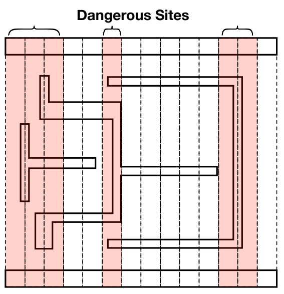


Fig. 3. An example of cell BUF_X8, where dangerous sites are labeled as red.

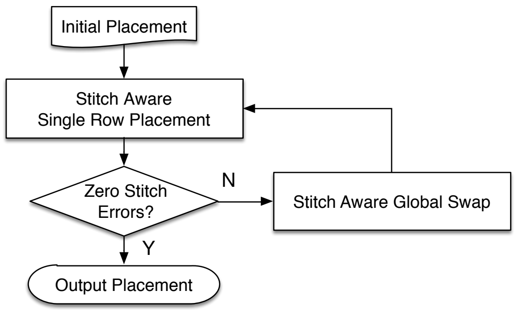


Fig. 4. Overall flow of our stitch aware detailed placement.

Problem 1 (Stitch Aware Detailed Placement)**.** *Given an initial detailed placement with the information of dangerous sites for each standard cell, we seek a legal placement to minimize the stitch errors and the sHPWL, simultaneously.*

III. Detailed Placement Algorithms

In this section we describe the details of our placement algorithms. As shown in Fig. 4, our framework mainly consists of two stages. In the first stage, single row based approach is applied to optimize wirelength and stitch errors optimally. If all stitch errors are removed successfully by this stage, we directly output placement solutions. Otherwise, in the second stage, cell swapping and movement are introduced to improve both wirelength and congestion. Note that the stitch error is considered through the whole flow.

A. Single Row Placement

As a powerful approach in detailed placement, single row based placement is widely studied in both conventional placement [10–12] and lithography aware application, such as triple patterning lithography (TPL) compliance [13–15]. If there are fixed macros in the layout, conventional single row algorithms (e.g. Abacus [12]) divide a row into several sub-rows. However, this strategy is not suitable for MEBL application, as the stitch lines are soft constraints rather than hard constraints. In TPL compliance, the main challenge lies in the distance between abutting cells, while the stitch errors in MEBL are not related to neighboring cells. In addition, in the single row algorithm proposed by [13], a graph based approach is applied to find optimal solution in $O(mnK)$. Here m is the site number in the row, n is the cell number, and K is the number of pre-coloring solutions for each cell. Usually m is a very large

TABLE I
NOTATIONS USED IN SINGLE ROW PLACEMENT

M	Maximum displacement for a cell.
p_i^0	Initial position of Cell c_i.
p_i	The position of Cell c_i, $p_i^0 - M \le p_i \le p_i^0 + M$.
$\alpha_i(p_i)$	solution of c_1 to c_i in which c_i is placed at p_i.
$t_i(p_i)$	The cost of best placement solution from c_1 to c_i in which c_i is placed at p_i.
$\gamma_i(p_i)$	The position of c_{i-1} in the optimal solution of c_1 to c_{i-1} in which c_i is at p_i.
$r_i(p_i)$	Whether the solution corresponding to $t_i(p_i)$ is inferior or not.
$cost_i(p_i)$	The cost of c_i when it is placed to p_i.
w_i	Width of Cell c_i.

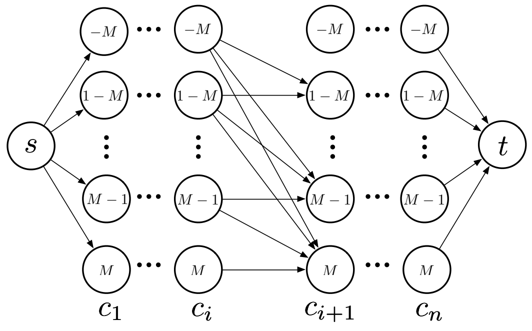


Fig. 5. Single row placement algorithm.

number, thus this algorithm may suffer from runtime for large size circuits.

In this paper we adapt a dynamic programming based algorithm [16] to solve single row detailed placement. Different from other techniques (e.g. [12]), it can naturally handle both hard constraints (fixed macros) and soft constraints (stitch errors). Each cell is associated with a movable range, which is usually a finite site candidates. The dynamic programming scheme is able to achieve optimal solution for combined cost functions, such as movement, wirelength and stitch errors. Note that comparing with [16], we significantly improve the runtime complexity while still maintaining the optimality.

For convenience, Table I lists some definitions used in the single row placement. The algorithm for single row placement is explained with a graph in Fig. 5. All candidate displacement values of a cell is listed as a column of nodes. There is an edge between two nodes if they can reach their displacement values without overlap. For example, the maximum displacement for cell c_i is M, so the displacement range for c_i is from $-M$ to M, the value of which is marked in the node. Each edge also contains a cost according to Eqn. (2). Two additional nodes, s and t, are inserted to the graph. The problem is stated as finding the path with lowest cost from node s to node t, which can be solved with dynamic programming.

The $cost_i(p_i)$ function in the experiments is as follows,

$$cost_i(p_i) = \tau \cdot WL(p_i) + \phi \cdot MOV(p_i) + \nu \cdot SP(p_i),$$
$$SP(p_i) = \begin{cases} 0, & \text{no stitch,} \\ \text{large number,} & \text{generate a stitch error,} \end{cases} \quad (2)$$

where WL denotes wirelength cost, MOV denotes movement, and SP denotes stitch error penalty. SP is set to a very large number when a stitch error is generated, e.g., half-perimeter of the layout. In our experiments, τ, ϕ, and ν are set to 10, 1, and 1. In legalization step, we simply set τ and ν to zero.

Given an ordered sequence of cells S, to calculate wirelength cost for cell c_i, we need to fix the positions for all other cells. The wirelength cost is determined by the bounding boxes of nets. But if cell c_i has connection to any cell in S, the wirelength cost for cell c_i cannot be determined since cells in S are not fixed. To handle this, we introduce the wirelength model in [17] which ensures the wirelength cost for cell c_i is independent to other cells in S, while the optimality of the solutions are maintained. It turns out that this wirelength model is equivalent to HPWL.

We can see that function $cost_i(p_i)$ is quite flexible, since we can include movement, wirelength and stitch errors. For hard constraints like fixed macros, we only need to set its maximum displacement M to zero. For soft constraints like stitch lines, additional cost is applied if a cell has overlap with them.

Lemma 1. *Algorithm shown in Fig. 5 is optimal for cost function in Eqn.* (2).

The proof is similar to that in [16], and is omitted here for brevity. The basic idea is that the optimal placement solution can be found through a shortest path from s to t, and all the positions of cells can be derived from the displacement values of corresponding nodes. Since the constructed graph is a directed acyclic graph, the shortest path can be calculated using topological traversal in $O(nM^2)$ steps, where n is the cell number in the row, and M is the maximum displacement for each cell.

B. An $O(nM)$ Pruning Algorithm

The runtime complexity of the above single row placement is $O(nM^2)$. When M is very large, the runtime becomes unacceptable. Here we propose a set of pruning techniques to achieve further speedup, while still keeping the optimality. In addition, we can theoretically prove that the runtime complexity can be improved from $O(nM^2)$ to $O(nM)$.

Algorithm 1 Single Row Placement with Pruning

```
Require: A set of ordered cells $c_1$ to $c_n$ of a row.
Ensure: All the cells in the set are placed subjecting to optimal objective function
 1: $t_1(p_1) \leftarrow cost_1(p_1), p_1 \in [p_1^0 - M, p_1^0 + M]$;
 2: $t_i(p_i) \leftarrow \infty, i \leftarrow 2\ to\ n, p_i \in [p_i^0 - M, p_i^0 + M]$;
 3: $r_i(p_i) \leftarrow 0, i \leftarrow 1\ to\ n, p_i \in [p_i^0 - M, p_i^0 + M]$;
 4: for each $c_i$, $i \leftarrow 2\ to\ n$ do
 5:     $N \leftarrow p_{i-1}^0 - M$;
 6:     for each $p_i \in [p_i^0 - M, p_i^0 + M]$ do
 7:         for each $p_{i-1} \in [N, p_{i-1}^0 + M]$ do
 8:             if $r_{i-1}(p_{i-1}) = 0$ then
 9:                 $cost \leftarrow t_{i-1}(p_{i-1}) + cost_i(p_i)$;
10:                 if $cost < t_i(p_i)$ then
11:                     $t_i(p_i) \leftarrow cost$;
12:                     $\gamma_i(p_i) \leftarrow p_{i-1}$;
13:                     $N \leftarrow p_{i-1}$;
14:                 else
15:                     break;
16:     Check inferior solutions and mark their $r_i(p_i)$ to 1;
17: $cost_{min} \leftarrow \infty$;
18: for $p_n \in [p_n^0 - M, p_n^0 + M]$ do
19:     if $t_n(p_n) < cost_{min}$ then
20:         $cost_{min} \leftarrow t_n(p_n)$;
21:         $P_n \leftarrow p_n$;
22: for $i \leftarrow n$ down to 2 do
23:     $P_{i-1} \leftarrow \gamma_i(p_i)$;
```

The details of our $O(nM)$ implementation is shown in Algorithm 1. The main difference between the problems in [13, 16] and our problem lies in the cost function. That is, the cost functions for a cell in the former problems depend on other cells, such as the distance or coloring cost between two abutting cells, while the cost defined in Eqn. (2) is only related to the cell itself; i.e., it is independent to any other cell. Due to the independence in the cost function, we can minimize the total cost with $O(nM)$ time complexity. Our speedup technique is **generic** that it can also be applied into conventional detailed placement and legalization with an objective like wirelength or movement.

Lemma 2. *Comparing two solutions $\alpha_i(p_i)$ and $\alpha_i(q_i)$, if $t_i(p_i) \geq t_i(q_i)$ and $p_i \geq q_i$, then $\alpha_i(p_i)$ is inferior to $\alpha_i(q_i)$*

Proof. Suppose cell c_i has two candidate positions p_i and q_i, where $p_i \geq q_i$ and $t_i(p_i) \geq t_i(q_i)$. Now consider any candidate position p_{i+1} for cell c_{i+1}. If cell c_i can be placed at p_i without overlapping with cell c_{i+1}, then q_i is also a legal position for cell c_i. We can always move cell c_i from p_i to q_i for better cost, because the total cost at cell c_{i+1} is the minimum value of $t_i(p_i) + cost_{i+1}(p_{i+1})$. Therefore, solution $\alpha_i(q_i)$ is better than $\alpha_i(p_i)$. □

Lemma 2 corresponds to line 8 and line 20 in Algorithm 1 where all inferior solutions are marked and skipped in the for loop. It implies that $t_i(p_i) < t_i(q_i)$ when $p_i > q_i$.

If p_{i-1} introduces overlaps between cell c_{i-1} and c_i, the cost is assigned to infinity. After skipping all inferior solutions, one should also note that in line 10, the condition $cost < t_i(p_i)$ is always satisfied when no overlapping occurs. The reason lies in that $t_{i-1}(p_{i-1})$ is decreasing w.r.t p_{i-1} according to Lemma 2 and $cost_i(p_i)$ does not change in the for loop from line 7 to 18. So the *else* condition in line 14 only happens when p_{i-1} results in overlaps, and we can break the loop under such a condition.

Lemma 3. *Let p^*_{i-1} be the optimal position of cell c_{i-1} when cell c_i is placed at p_i, and q^*_{i-1} be the optimal position of cell c_{i-1} when cell c_i is placed at q_i. If $q_i \geq p_i$, then $q^*_{i-1} \geq p^*_{i-1}$.*

Proof. For a legal position p_i of cell c_i, to minimize $t_i(p_i)$, we need to find the smallest $t_{i-1}(p_{i-1})$ for all possible p_{i-1}, because $cost_i(p_i)$ has been determined by p_i. Let P_{i-1} be the set of all legal values of p_{i-1} and Q_{i-1} denote all possible values of q_{i-1}. Suppose p^*_{i-1} is the best position for cell c_{i-1} when c_i is placed at p_i and q^*_{i-1} is the best position for cell c_{i-1} when c_i is placed at q_i. P_{i-1} and Q_{i-1} should share the same left boundary l. Let P^r_{i-1} be the right boundary of set P_{i-1} and Q^r_{i-1} be the right boundary of set Q_{i-1}. P^r_{i-1} is no larger than Q^r_{i-1}, as $q_i \geq p_i$. In other words, we have $P_{i-1} \subseteq Q_{i-1}$. The relationship can be rewritten as,

$$\begin{aligned} &P_{i-1} = \{p_{i-1} \mid l \leq p_{i-1} \leq P^r_{i-1}, p_{i-1} \in \mathrm{Z}\}, \\ &Q_{i-1} = \{q_{i-1} \mid l \leq q_{i-1} \leq Q^r_{i-1}, q_{i-1} \in \mathrm{Z}\}, \\ &P^r_{i-1} \leq Q^r_{i-1}. \end{aligned}$$

If q^*_{i-1} lies in the range between l and P^r_{i-1}, it must be equal to p^*_{i-1}; otherwise, it is equal to some value between P^r_{i-1} and Q^r_{i-1}. Hence, $q^*_{i-1} \geq p^*_{i-1}$. □

Lemma 3 corresponds to line 5, 7 and 13 in Algorithm 1. After computing the optimal solution for cell c_i at position $p_i = q$, We can start p_{i-1} from N (line 5 of Algorithm 1) instead of $p^0_{i-1} - M$ to find an optimal solution for a value $p_i > q$.

The above analysis guarantees the optimality of Algorithm 1. Compared with previous $O(M^2)$ algorithms in [13, 16], Algorithm 1 changes the complexity of the local search (from line 6 to line 19) to $O(M)$. Line 20 also takes $O(M)$ time to mark all inferior solutions. The runtime complexity of Algorithm 1 is $O(nM)$.

It should be noted that our pruning algorithm is flexible to any cost function $cost_i(p_i)$ as long as it only depends on p_i itself. That is, it can be applied to speed-up the conventional single row detailed placement problems [10–12].

C. Stitch Aware Global Swap

In this step, the main objective is to optimize regions that contain cells involved in stitch errors. After the optimization of single row placement, most stitch errors have been resolved. The remaining ones usually appear in highly congested placement bins. Therefore, we only try to move cells in such bins to alleviate the congestion and meanwhile reduce wirelength.

Due to the congestion of these regions, it is difficult to resolve them with local perturbation such as reordering or sliding window. Thus global swap [17, 18] is adopted where cells are allowed to move anywhere within the displacement constraints. Generalized swap not only enables swapping with cells but also white spaces, which integrates both swapping and moving strategies. The basic procedure for cell swap is iteratively repeating the following three steps: (1) Select a source cell to swap; (2) Identify optimal region for source cell; (3) Find the best cell or white space to swap with the source cell in the optimal region.

In our implementation, we set the score function for swap as follows,

$$score(c_i, c_j) = \Delta sHPWL - \lambda \cdot P_{ds} - \mu \cdot P_{ov}, \qquad (3)$$

where $\Delta sHPWL$ indicates sHPWL improvement, P_{ds} indicates the penalty for density increase of dangerous sites, and P_{ov} is overlap penalty. Suppose cell c_i is in bin B_i and cell c_j belongs to bin B_j. The area of both bins is A_b. We define the *density of dangerous sites* as the number of dangerous sites over total amount of available sites in a bin. If a bin has overlap with any stitch line, we account only 70% of its total sites as available. Let $D_{ds}(i)$ denote the density of dangerous sites in bin B_i before swap and $D'_{ds}(i)$ denote the density of dangerous sites in bin B_i after swap. Then we can define P_{ds} with the following equation,

$$P_{ds} = \max(0, |D'_{ds}(i) - D'_{ds}(j)| - |D_{ds}(i) - D_{ds}(j)|) \cdot A_b \quad (4)$$

The overlap penalty is the area difference between the source cell and target cell or white space. If the target white space is larger than the source cell, overlap penalty is zero. To achieve an equivalent numeric scale to wirelength cost, P_{ds} and P_{ov} are divided by site half-perimeter in the implementation. In this way, all the costs have the same unit as distance. λ and μ are set to 100. Only swapping attempt with best positive scores is accepted.

The scoring scheme proposed in Eqn. (3) aims for balancing the density of cells and dangerous sites while improving wirelength. Although the penalty from ABU density is able to handle global density distribution, local control is necessary to avoid extremely dense regions. Furthermore, it is easier for a congested region with very few dangerous sites to find a stitch-error-free solution than that with a lot of dangerous sites. Thus we introduce P_{ds} as the additional penalty for such kind of regions. Since row-based legalization engine is applied, the height of bins for P_{ds} is set to row height.

Overlap penalty is introduced to control the efforts during legalization. High legalization efforts will incur large displacement for some cells and thereby large wirelength degradation. Hence, after every 5000 swaps, legalization algorithm will be performed to remove overlaps. Legalization algorithm is based on single-row placement (Section A) with minimum movement as an objective.

We observe that the runtime for global swap is highly related to the complexity of score function. Considering that wirelength is included in the calculation, it will be very slow to query the bounding box of large nets. Thus we develop a data structure in which pins of a net are stored as an ordered sequence according to pin positions. Cells in a row is kept in a linked-list [18] for fast cell swap and movement.

Usually a cell is connected to limited number of nets, thus its degree can be treated as constant. Using the data structures above, it only takes constant time to query the bounding box and $O(\log e)$ to update cell position in a net with e pins. Since score calculation happens much more frequent than actual cell swap or movement, faster score calculation helps to reduce overall runtime. Let k be the number of swapping candidates for a cell c_i, we can achieve $O(k)$ time complexity for score calculation and $O(\log e_{max})$ for cell position update if a swap or movement is accepted, where e_{max} is the maximum e of nets connected to cell c_i.

IV. Experimental Results

Our algorithms were implemented in C++ and tested on a 3.40 GHz Linux machine with 32 GB memory. Since traditional academic placement benchmark suites has no intra-cell wire information, we integrated the NanGate $15nm$ standard cell library [19] into ICCAD 2014 placement benchmarks [9]. ICCAD 2014 placement contest defines two maximum displacement values for each benchmark, and we choose the smaller ones for less perturbation to the original placements. We applied a state-of-the-art detailed placer, *RippleDp* [20], to generate the initial placement solutions. We scaled the bin dimensions for ABU density analysis from the ICCAD 2014 benchmarks, so most generated test cases match to the number of bins in the original ones. We pre-computed dangerous sites for all standard cells in the library, which was served as input to our placer. We set the stripe width of each single beam to $50\mu m$.

The metrics of the new benchmarks are shown in Table II, where columns "#cells" and "#nets" list the total cell number and net number, respectively. Besides, columns "#blk", "d_t" and "Disp." represent the blockage (fixed macro) number, the target density, and the maximum displacement in um. Note

TABLE II
Benchmarks for Stitch Aware Placement

Design	#cells	#nets	#blk	Density	d_t	Disp.
vga_lcd	165K	165K	0	68.94%	70%	10
b19	219K	219K	0	44.85%	70%	20
leon3mp	649K	649K	0	72.02%	75%	30
leon2	794K	795K	0	84.19%	90%	40
mgc_edit_dist	131K	133K	13	67.26%	70%	30
mgc_matrix_mult	155K	159K	16	59.31%	65%	30
netcard	959K	961K	12	66.29%	70%	50

that test cases *mgc_edit_dist*, *mgc_matrix_mul* and *netcard* contain mixed-sized cells.

Table III lists the performance of our placer at different optimization stages. The initial placement solutions (column "Init.") are generated by a traditional detailed placer, RippleDp [20], which aims at minimizing wirelength. As the state-of-the-art detailed placer, RippleDp can produce very high quality placement solutions in terms of both HPWL and sHPWL. Here we set displacement constraint to be a very large number so that RippleDp can produce converged results. Column "**SR**" stands for single row placement, while column "**Full Flow**" denotes the whole flow combining global swap and single row placement. To evaluate the effectiveness of our algorithms, following metrics are introduced. **HPWL** stands for half perimeter wirelength which is used as a metric for wirelength. **ST#** represents the number of cells that contains stitch errors. It is measured by how many dangerous sites are covered by the beam boundaries. Placement solutions with high congestion are not desired, so we introduce **sHPWL** as discussed in Section II. When measuring **Runtime**, which is the CPU run time in seconds, single thread is applied for consistency of results.

From Table III we can see that, with certain displacement constraints, the proposed single row placement can achieve very good efficacy in stitch error cancellation. That is, 99.9% of the initial stitch errors are removed. Meanwhile, an average of 0.19% HPWL improvement and slight sHPWL increase are observed. However, for some corner cases, such as *leon2* and *netcard*, the single row placement is not powerful enough due to the movement constraints from blockages or congestions. Therefore, global swap is introduced as a follow-up optimization step, and the corresponding results are shown in the last column. We can see that swapping cells between rows improves congestion in dense regions and optimize wirelength. By applying global swap together with single row algorithm, we are able to achieve zero stitch errors for all test cases. As only small number of bins are considered for global swap, the runtime overhead can be neglected. Small changes in HPWL and sHPWL also indicate that the algorithm produces little perturbation to initial placement.

It should be noted that the runtime of single row placement in Table III for case *netcard* is very close to that of *leon2*, while the former has much larger cell number. The reason lies in those blockages in *netcard*. That is, the runtime of single row placement is not only related to the number of cells, but also the amount of maximum displacement. Blockages have zero maximum displacement. So during the propagation of candidate solutions in the dynamic programming process, many infeasible solutions are automatically pruned. Therefore, the solution space has been significantly reduced and as a consequence, the

TABLE III
RESULT COMPARISON AMONG DIFFERENT APPROACHES.

Design	Init.			SR				Full Flow			
	HPWL ($\times 10^6$)	sHPWL ($\times 10^6$)	ST #	ΔHPWL (%)	ΔsHPWL (%)	ST #	Runtime (s)	ΔHPWL (%)	ΔsHPWL (%)	ST #	Runtime (s)
vga_lcd	1.42	1.87	1266	-0.28	+0.36	0	7.70	-0.28	+0.36	0	7.74
b19	0.97	1.14	1435	-0.25	-0.00	0	10.54	-0.25	-0.00	0	10.74
leon3mp	5.34	6.84	6474	-0.32	-0.14	0	33.36	-0.32	-0.14	0	33.59
leon2	13.09	14.49	8172	-0.09	+0.12	1	42.24	-0.10	+0.11	0	49.90
med	1.52	1.88	864	-0.14	+0.18	0	6.05	-0.14	+0.18	0	6.11
mmm	0.91	1.13	1117	-0.17	-0.13	0	7.22	-0.17	-0.13	0	7.28
netcard	14.57	20.19	7789	-0.11	+0.06	21	44.68	-0.10	+0.08	0	50.02
avg.	5.40	6.79	3873	-0.19	+0.06	3	21.68	-0.19	+0.07	0	23.62
ratio	-	-	**1**	-	-	**0.001**	1.00	-	-	**0**	1.09

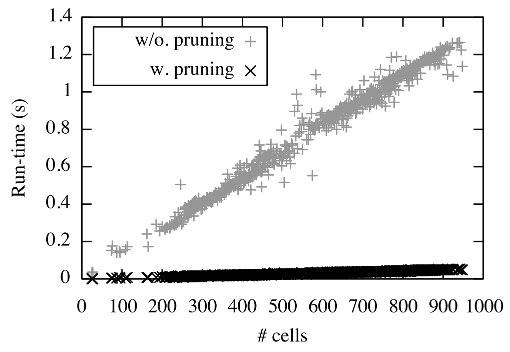


Fig. 6. Comparison on algorithm scalability.

best solution is found in shorter time.

Fig. 6 compares the runtime difference for variant amounts of cells in a row between whether applying pruning techniques or not. The data is directly collected from benchmarks in Table II and the runtime values of rows with the same number of cells are averaged. We can see the runtime grows linearly with the problem size and the difference in the slopes shows that pruning techniques effectively drop runtime. On average, the $O(nM)$ pruning technique can provide around 30× speedup without any loss of optimality.

V. CONCLUSION

This work develops the first placement framework considering e-beam stitch errors during detailed placement stage. A linear-time single row placement algorithm is proposed with highly-adaptable objective functions. Experimental results show its effectiveness in stitch cancellation while maintaining wirelength and congestion. With the collaboration of stitch aware post-placement optimization such as [5], better manufactorability can be achieved. In addition, our high performance pruning technique can be natually embedded into existing physical design flow with different metrics (e.g., wirelength, routability, or congestion).

ACKNOWLEDGMENT

This work is supported in part by NSF and SRC. Thanks to William Chou and Prof. F. Y. Young for the updated version of RippleDp.

REFERENCES

[1] D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Transactions on CAD*, vol. 32, no. 10, pp. 1453–1472, 2013.

[2] B. J. Lin, "Future of multiple-e-beam direct-write systems," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)*, vol. 11, no. 3, pp. 033 011–1, 2012.

[3] C. Van den Berg, G. De Boer, S. Boschker, E. Hakkennes, G. Holgate, M. Hoving, R. Jager, J. Koning, V. Kuiper, Y. Ma *et al.*, "Scanning exposures with a MAPPER multibeam system," in *Proc. of SPIE*, vol. 7970, 2011.

[4] M. A. McCord, P. Petric, U. Ummethala, A. Carroll, S. Kojima, L. Grella, S. Shriyan, C. T. Rettner, and C. F. Bevis, "REBL: design progress toward 16 nm half-pitch maskless projection electron beam lithography," in *Proc. of SPIE*, vol. 832311, 2012.

[5] S.-Y. Fang, I.-J. Liu, and Y.-W. Chang, "Stitch-aware routing for multiple e-beam lithography," in *DAC*, 2013, pp. 25:1–25:6.

[6] K. Suzuki, T. Fujiwara, K. Hada, N. Hirayanagi, S. Kawata, K. Morita, K. Okamoto, T. Okino, S. Shimizu, and T. Yahiro, "Nikon EB stepper: its system concept and countermeasures for critical issues," in *Proc. of SPIE*, vol. 3997, 2000.

[7] D. Dougherty, R. Muller, P. Maker, and S. Forouhar, "Stitching-error reduction in gratings by shot-shifted electron-beam lithography," *Journal of Lightwave Technology*, vol. 19, no. 10, pp. 1527–1531, oct 2001.

[8] J. Albert, S. Theriault, F. Bilodeau, D. Johnson, K. Hill, P. Sixt, and M. Rooks, "Minimization of phase errors in long fiber bragg grating phase masks made using electron beam lithography," *IEEE Photonics Technology Letters*, vol. 8, no. 10, pp. 1334–1336, oct. 1996.

[9] M.-C. Kim, J. Hu, and N. Viswanathan, "ICCAD-2014 CAD contest in incremental timing-driven placement and benchmark suite," in *ICCAD*, 2014, pp. 361–366.

[10] U. Brenner and J. Vygen, "Faster optimal single-row placement with fixed ordering," in *DATE*, 2000, pp. 117–121.

[11] A. B. Kahng, I. L. Markov, and S. Reda, "On legalization of row-based placements," in *GLSVLSI*, 2004, pp. 214–219.

[12] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: fast legalization of standard cell circuits with minimal movement," in *ISPD*, 2008, pp. 47–53.

[13] B. Yu, X. Xu, J.-R. Gao, and D. Z. Pan, "Methodology for standard cell compliance and detailed placement for triple patterning lithography," in *ICCAD*, 2013, pp. 349–356.

[14] J. Kuang, W.-K. Chow, and E. F. Y. Young, "Triple patterning lithography aware optimization for standard cell based design," in *ICCAD*, 2014, pp. 108–115.

[15] T. Lin and C. Chu, "TPL-aware displacement-driven detailed placement refinement with coloring constraints," in *ISPD*, 2015, pp. 75–80.

[16] T. Taghavi, C. Alpert, A. Huber, Z. Li, G.-J. Nam, and S. Ramji, "New placement prediction and mitigation techniques for local routing congestion," in *ICCAD*, 2010, pp. 621–624.

[17] M. Pan, N. Viswanathan, and C. Chu, "An efficient and effective detailed placement algorithm," in *ICCAD*, 2005, pp. 48–55.

[18] S. Popovych, H.-H. Lai, C.-M. Wang, Y.-L. Li, W.-H. Liu, and T.-C. Wang, "Density-aware detailed placement with instant legalization," in *DAC*, 2014, pp. 122:1–122:6.

[19] "NanGate FreePDK15 Open Cell Library," http://www.nangate.com/?page_id=2328, 2015.

[20] W.-K. Chow, J. Kuang, X. He, W. Cai, and E. F. Young, "Cell density-driven detailed placement with displacement constraint," in *ISPD*, 2014, pp. 3–10.