Eliminating Minimum Implant Area Violations With Design Quality Preservation

Eunsol Jeong, Taewhan Kim, *Senior Member, IEEE*, and Heechun Park

***Abstract*— Minimum implant area (MIA) violation has emerged in the sub-micrometer technology which requires a certain amount of threshold voltage (V_t) area for the fabrication. Elimination of MIA violations in the sign-off layout thus becomes an inevitable task for a high-performance multiple-V_t design. Conventional approaches as well as the previous efforts to remove MIA violations bring severe defects to the final design in that locally moving cells or reassigning V_ts make the timing constraints unsatisfied or power consumption to be exploded. In this article, we propose a comprehensive MIA violation removal algorithm that fully and systematically controls the timing budget and power overhead with three sequential steps: 1) removing intra-row MIA violations by V_t reassignment under timing preservation and minimal power increments; 2) removing inter-row MIA violations with a theoretically optimal V_t reassignment while satisfying timing constraints; and 3) refining V_t reassignment to recover the power loss without violating both MIA constraints and timing closure. Moreover, we introduce a preprocessing algorithm at the preroute stage to remove a huge amount of MIA violations in advance for an additional runtime reduction without design quality degradation. Experiments through benchmark circuits show that our proposed approach completely resolve MIA violations while ensuring no timing violation and using 34.6% less power overhead on average than the conventional approaches and previous works. In addition, our preprocessing step reduces 45%–88% of MIA violations before the routing stage, which incurs 41% faster MIA removal on average in the final stage with similar design quality.**

Manuscript received 10 June 2022; revised 6 September 2022 and 12 October 2022; accepted 20 November 2022. Date of publication 8 December 2022; date of current version 26 April 2023. This work was supported in part by the Samsung Electronics Company Ltd., under Grant IO201216-08205-01; in part by the FOUNDRY under Grant 202010DD003F; in part by the Electronic Design Automation (EDA) Cluster Project; in part by the National Research Foundation of Korea (NRF) grant funded by the Korean Government [Ministry of Education, Science and Technology (MEST)] under Grant 2020R1A4A4079177 and Grant 2021R1A2C2008864; in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2021R1I1A1A01049399; in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean Government [Ministry of Science and ICT (MSIT)] under Grant 2021-0-00754; in part by the Software Systems for Artificial Intelligence (AI) Semiconductor Design; in part by the National Research and Development Program through the NRF funded by MSIT under Grant 2020M3H2A1078119; and in part by the BK21 Four Program of the Education and Research Program for Future ICT Pioneers, Seoul National University, in 2021. *(Corresponding author: Heechun Park.)*

Eunsol Jeong and Taewhan Kim are with the Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea.

Heechun Park is with the School of Electrical Engineering, Kookmin University, Seoul 02707, South Korea (e-mail: phc@kookmin.ac.kr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TVLSI.2022.3225551.

Digital Object Identifier 10.1109/TVLSI.2022.3225551

***Index Terms*— Intra-row minimum implant area (MIA) violations, mixed integer-linear programming (MILP), multi-V_t designs.**

I. Introduction

MULTI-THRESHOLD voltage (V_t) designs are widely employed in implementing power-aware high-performance chips [1] to reduce leakage power and in the meantime satisfy timing constraints. Low-V_t (LVT) standard cells drive signal faster but consume more leakage power, and regular/high-V_t (RVT/HVT) cells operate relatively slower and less leaky. LVT cells are used on timing critical paths to achieve short path delay, and the other (RVT/HVT) cells are located on noncritical paths to save power consumption at the expense of delay increase. Designers leverage LVT/RVT/HVT cells to enhance the design quality, i.e., less leakage power while satisfying timing constraints.

As the technology node scales down to 22 nm and below, manufacturing restrictions limit the minimum area of a V_t island in the implant region [2], called minimum implant area (MIA) constraint. MIA violations are classified into two types, namely intra-row and inter-row MIA violations. Fig. 1 illustrates both types of violations. An intra-row MIA violation [see Fig. 1(a)] happens when the area of a V_t island is smaller than the limit of the MIA. It means that in the chip implementations with row-based cell placement, an implant width should be longer than the minimum implant width limit (W) to satisfy the intra-row MIA constraint. An inter-row MIA violation [3] [see Fig. 1(b)] takes place when the same V_t on adjacent rows form a staircase and its abutting width is shorter than W.

MIA violations were insignificant in the traditional process nodes, as the violated cases are rare and can be simply resolved by inserting filler cells with the same V_t to the violated regions. However, in the advanced process nodes with 22 nm and below, the number of intra-row MIA violations has sharply increased, and it is now almost impossible to resolve the violations by solely relying on the filler cell insertion. Furthermore, the emergence of inter-row MIA violations on further advanced technology below 10 nm has made resolving or avoiding both intra- and inter-row MIA violations a nontrivial task.

Conventional approaches have tried to locally move cells or reassign V_t of some cells in a way to resolve the MIA violations. However, these approaches are missing the consideration on the design quality, especially power consumption and timing constraints. Changing standard cells to lower V_t cells

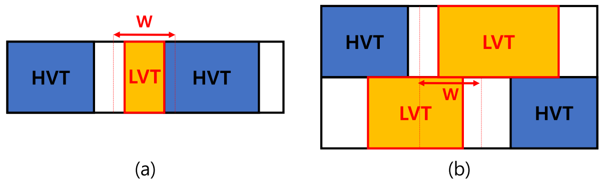

Fig. 1. Example of two types of MIA violation in row-based cell placement. (a) Intra-row MIA violation. (b) Inter-row MIA violation.

(e.g., HVT $\rightarrow$ RVT/LVT, RVT $\rightarrow$ LVT) for MIA violations will increase the total power consumption of the design. Replacing with higher V_t cells (e.g., LVT $\rightarrow$ RVT/HVT, RVT $\rightarrow$ HVT) or shifting cell locations incur drastic changes on data propagation delays for the timing critical paths, which have possibilities to generate negative slacks, i.e., violating timing constraints. To resolve the timing violations, additional cell displacements or repeater insertions are required, which may bring up new MIA violation spots.

In this work, we propose a systematic approach that fully controls the timing budget during the removal of MIA violations. Precisely, our solution consists of three sequential steps: 1) performing critical path aware cell selection for V_t reassignment to fix the intra-row MIA violations while considering the timing constraint and minimal power increments; 2) performing a theoretically optimal V_t reassignment to fix the inter-row MIA violations while satisfying both intra-row MIA and timing constraints; and 3) refining V_t reassignment to further reduce the power consumption while meeting both MIA constraints as well as timing constraints. The novelty of our MIA violation removal algorithm is that we keep track of the timing preservation and power increments throughout the flow to maintain the design quality when we modify V_ts to remove violations.

In addition, we introduce a preprocessing algorithm at the preroute stage to reduce the occurrence of MIA violations beforehand. While our MIA violation removal algorithm is conducted at the final stage and thus is only allowed a conservative design modification to preserve timing closure, our preprocessing step refines the design more aggressively (e.g., exploit cell movement) to reduce a huge amount of problem space for the MIA violation removal algorithm. Its adverse effect on the timing closure is trivial and can be countervailed in the final timing closure stage.

II. Preliminaries and Related Works

Fig. 2 illustrates the possible solutions to fix different MIA violations. Filler cell insertion [see Fig. 2(a)] is the most common solution, which adds filler cells with appropriate V_t to whitespaces to expand the V_t island width beyond W and solve intra-row MIA violations. A filler cell does not consume any power and the V_t of it does not incur any design cost. Thus, the ideal solution to resolve all MIA violations is to only apply filler cell insertions, which do not touch the design quality. Synopsys IC Compiler II (ICC2) provides a basic filler cell insertion flow according to the V_t of both sides of the target whitespace and user-defined rules.

However, in the advanced technology nodes, as the standard cells become smaller and inter-row MIA violation becomes

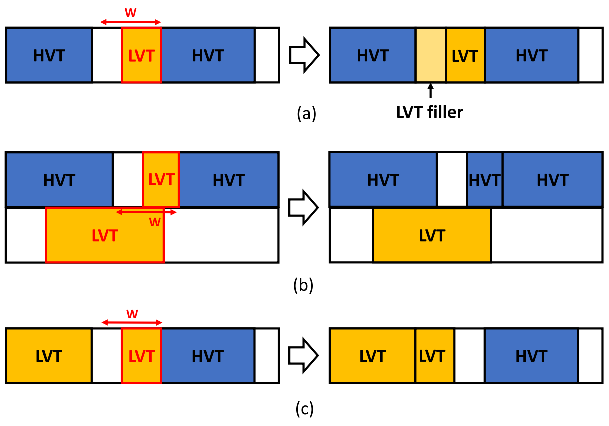

Fig. 2. Examples of fixing MIA violation through (a) filler cell insertion, (b) V_t reassignment, and (c) cell displacement.

non-trivial, filler cell insertion is no longer enough to eliminate all MIA violations, as shown in Table I in Section V-A. In such an environment, we can apply V_t reassignment [see Fig. 2(b)] and cell displacement [see Fig. 2(c)] to remove small V_t island or short inter-row V_t abutting width. These two methods modify the internal design feature (i.e., V_ts or locations of standard cells), and thus contain the riskiness of design degradation in terms of timing closure and power consumption.

V_t assignment is a well-studied method to optimize the performance by improving timing closure and reducing power consumption (e.g., [4], [5]). There are several previous efforts to resolve MIA violations. Kahng and Lee [2] proposed a heuristic algorithm of row-based V_t reassignment and cell displacement to remove MIA violations by rearranging V_t islands to increase implant area. Later, Lei et al. [6] and Mak et al. [7] formulated the problem as mixed integer-linear programming (MILP) model to solve both intra- and inter-row MIA violations simultaneously. They claimed that their V_t reassignments maintain timing closure from the baseline design (i.e., before applying the algorithms) by only allowing to reassign the standard cells' V_ts to lower V_ts, which means the changed cells always have shorter propagation delay (i.e., become faster). However, they did not show a thorough analysis about the effects of cell displacement, as it requires to reroute the wires connected to the displaced cells and thus affects more on the timing closure. Since the impact of rerouting on the path delay becomes more significant in sub-10 nm technology [8], even a minimized cell displacement as in [7] will not guarantee the same timing preservation as the baseline design. Moreover, V_t reassignment only in the direction of lowering V_t incurs an excessive power overhead.

Prior works for MIA violation removal at the preroute stage (e.g., placement) also exist. Tseng et al. [9] proposed a cluster-based detailed placement flow to cluster cells with identical V_t. Han et al. [3] proposed a MILP-based algorithm to fix all MIA constraints in the detailed placement. Chen et al. [10] and [11] proposed a two-stage method to resolve the MIA problem in mixed-cell-height-based design at the placement and legalization stage. All these solutions are conducted at the placement stage, which has more freedom in modifying cell V_ts or locations. However, their main target

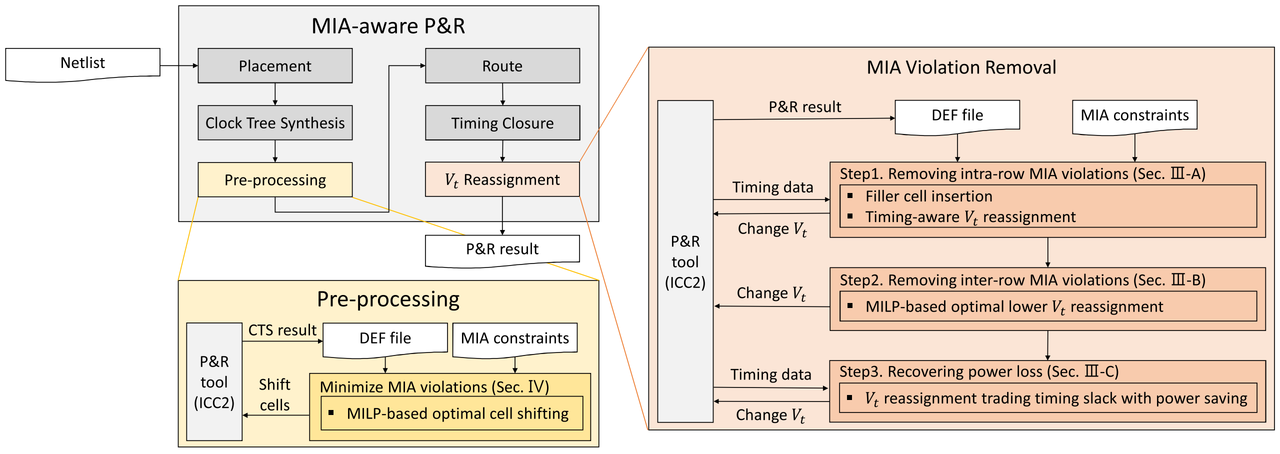
MIA-aware P&R
Netlist
Placement
Clock Tree Synthesis
Pre-processing
Route
Timing Closure
V_t Reassignment
P&R result
Pre-processing
P&R tool (ICC2)
CTS result
DEF file
MIA constraints
Shift cells
Minimize MIA violations (Sec. IV)
MILP-based optimal cell shifting
MIA Violation Removal
P&R result
DEF file
MIA constraints
Timing data
Change V_t
Step1. Removing intra-row MIA violations (Sec. Ⅲ-A)
Filler cell insertion
Timing-aware V_t reassignment
Step2. Removing inter-row MIA violations (Sec. Ⅲ-B)
MILP-based optimal lower V_t reassignment
Step3. Recovering power loss (Sec. Ⅲ-C)
V_t reassignment trading timing slack with power saving

Fig. 3. Block diagram of our proposed MIA violation removal flow (expanded from [12]).

metric was the half-perimeter wirelength (HPWL) from the placement results, which is not an ultimate value for design evaluation. They did not take the timing closure and the power consumption of the post-route design layout into account, as well as the effects of inserted or displaced cells during the post-placement stages (e.g., buffers from CTS and timing closure stages).

III. Proposed MIA Violation Removal Algorithm

In this section, we introduce our MIA violation removal algorithm at the final stage. The right side in Fig. 3 shows the overall flowchart with three steps. Step 1 eliminates all the intra-row MIA violations by heuristically performing V_t reassignment while minimizing the amount of the power increase as well as preserving the timing constraint. Step 2 is then conducted to eliminate all inter-row MIA violations while not violating the timing and intra-row MIA constraints met in Step 1 by formulating the inter-row MIA violation problem into a MILP model to find a solution of V_t reassignments to lower V_ts that satisfies zero inter-row MIA violation with the least power increments. Step 3 recovers the power loss caused by the V_t reassignment in Step 2 by iteratively performing V_t reassignment to higher V_ts for standard cells on the noncritical paths while not violating the MIA constraints as well as timing constraints met in Step 2. Finally, we obtain an MIA-violation-free design with preserved timing closure and only a little amount of power increments.

A. Step 1: Elimination of Intra-Row MIA Violations

The first step is to eliminate all intra-row MIA violation with filler cell insertions and timing-aware V_t reassignments. We do not exploit cell displacement for our MIA violation removal algorithm, as it requires an additional rerouting stage that incurs a huge side effect on the timing preservation. We also provide a speedup technique to accelerate the process of the V_t reassignment.

1) Filler Cell Insertion: In the beginning, we use filler cell insertion to remove intra-row MIA violation on the narrow (i.e., width $< W$) standard cells as much as possible. Fig. 4

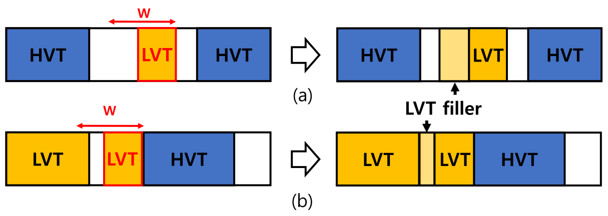

Fig. 4. Cases of filler cell insertion with V_t assignment. (a) Make the V_t island larger than W. (b) Merge V_t islands.

shows two different cases of intra-row MIA violations that can be removed by filler cell insertion. When a narrow cell has enough whitespaces for filler cells to solve its violation, we add filler cell(s) with the same V_t to make the V_t island has the same width as W [e.g., Fig. 4(a)]. Note that we add filler cells to a minimum amount and leave whitespaces as much as possible, instead of covering all whitespaces in Fig. 4(a) with filler cells, to fully utilize the remaining whitespaces in our inter-row MIA violation removal step (Step 2, details in Section III-B). When the amount of whitespace is not enough but the next cell has the same V_t, we can also solve the violation by inserting filler cell(s) into the whitespace to make the merged V_t island with two cells and a filler cell in between them [e.g., Fig. 4(b)].

2) Timing-Aware V_t Reassignment: We resolve the remaining intra-row MIA violations with timing-aware V_t assignment. For each of the MIA violated cells, we reassign V_t of the cell itself or one of its neighboring cells to enlarge the V_t island and remove the violation. In the meantime, we trace the timing critical path of the V_t reassigned cell (say *target cell*) and compensate for the V_t change by replacing another cell in the path (say *offset cell*) with a lower/higher V_t cell if the target cell was changed to a higher/lower V_t. While prior works (e.g., [2], [7]) only allowed the target cell V_t to move downward to preserve timing at the cost of power loss, we make it possible to assign higher V_t to the target cell followed by lowering an offset cell V_t to preserve timing and reduce power increments at the same time. This can be applied in the opposite way, i.e., lower V_t to the target cell and higher V_t to an offset cell, utilizing the increased positive slack

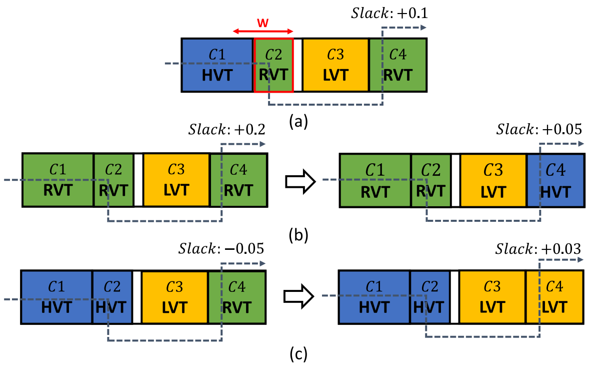

Fig. 5. Illustration of intra-row MIA violation removal. (a) Intra-row MIA violated cell ($C2$) without enough whitespaces. (b) Solution with assigning lower V_t to $C1$. (c) Solution with assigning higher V_t to $C2$.

by "faster" target cell onto the "slower" offset cell thereby reducing the total power consumption.

Fig. 5 shows an example of our timing-aware V_t reassignment, in which $C2$ is violating the intra-row MIA constraint. There are two ways of resolving it: 1) lower V_t (HVT $\rightarrow$ RVT) to $C1$ [see Fig. 5(b)] or 2) higher V_t (RVT $\rightarrow$ HVT) to $C2$ [see Fig. 5(c)]. For case 1, $C1$ is the target cell and its V_t became lower to remove violation from $C2$. This V_t reassignment also increases the positive slack of timing critical path through $C1$, say from 0.1 to 0.2. Then, we trace the timing critical path of $C1$ and find an offset cell candidate (e.g., $C4$) that is possible to be assigned a higher V_t without violating intra-row MIA constraint and preserve timing (i.e., slack > 0). For case 2, the target cell $C2$ is assigned to a higher V_t. This incurs the decrease on the path slack through $C2$, which become negative and is violating timing closure. We then find an offset cell in the path (e.g., $C4$) and assign a lower V_t to induce shorter path delay and larger slack without violating intra-row MIA constraint.

Algorithm 1 summarizes our timing-aware V_t reassignment algorithm for an intra-row MIA violating cell c_v. It is operated through all violating cells to find a full solution that all intra-row MIA violations are cleared. Each of the loops in the algorithm is explained in detail below.

a) Collect timing data: For an intra-row MIA violated cell (c_v), we firstly find all violation-free V_t combinations for c_v and its neighboring cells (C_n) (Line 1). For each case in the combination set (Comb$_v$), we trace through the timing critical path of the target cell (c_{targ}) and find candidates of the offset cell (c_{offset}) if its V_t can be shifted in the opposite direction to that of c_{targ} for timing and power compensation. We then collect the timing data (i.e., delay and current slack) for the offset cell candidates in T, and store the V_t assignment of c_{offset} in the queue of solutions S (Lines 4–9).

b) Prune invalid solutions: From the collected solutions, we prune out the invalid cases that incur extra intra-row MIA violation by c_{offset} or worse critical path slack after V_t updates (slack$_{\text{new}}$) than a constraint value (slack$_{\text{worst}}$) (Lines 11–15). While the occurrence of extra violation is easy to find, path slack comparison has to go through the P&R tool by reassigning V_ts and updating the timing information of the entire design (e.g., `update_timing` in Synopsys ICC2) for

Algorithm 1 Step 1: Timing-Aware V_t Reassignment for an Intra-Row MIA Violation

```
Input: $c_v$ (intra-row MIA violated cell), $C_n$ (neighboring cells of $c_v$), $slack_{worst}$ (worst slack)
1:  $Comb_v \leftarrow$ Violation-free $V_t$ combinations of $c_v$ and $C_n$
2:  $T, S \leftarrow \{\}$
3:  for $case_v$ in $Comb_v$ do                         ▷ Collect timing data
4:      for $c_{path}$ in $path(c_{targ})$ do
5:          if $c_{path}$ $V_t$ shift is in the opposite direction then
6:              $c_{offset} \leftarrow c_{path}$
7:              Update $T$ & $S$ with $c_{offset}$
8:          end if
9:      end for
10: end for
11: for $s$ in $S$ do                                   ▷ Prune invalid solutions
12:     Calculate $slack_{new}$
13:     if $s$ incurs violation or $slack_{new} < slack_{worst}$ then
14:         $S \leftarrow S - \{s\}$
15:     end if
16: end for
17: Sort $S$ in ascending order of estimated power overhead
18: $i \leftarrow 0$                                      ▷ $V_t$ reassignment
19: while $i < k$ do
20:     Update $V_t$ according to the top $i$'th solution in $S$
21:     Update timing with P&R tool
22:     if slack of $path(c_{targ}) < 0$ then
23:         return
24:     end if
25:     $i \leftarrow i + 1$
26: end while
```

accuracy, which is too time consuming. Instead, we calculate path slack referring to the V_t change without the P&R tool, and use the P&R tool only in the last loop (V_t reassignment). We first calculate the estimated delay change by V_t shift of cell c (Δd_c) according to the original cell delay and the common delay ratio according to the V_t. Then, we calculate the updated slack from the V_t change in a conservative manner as below

$$\text{slack}_{\text{new}} = \begin{cases} \text{slack}_{\text{org}} + (1-\alpha)|\Delta d_c|, & \text{if } \Delta d_c < 0 \\ \text{slack}_{\text{org}} - (1+\alpha)\Delta d_c, & \text{otherwise} \end{cases} \quad (1)$$

where α (0.0–0.3) is a tunable parameter for pessimistic timing estimation. In this way, shortened cell delay by lowered V_t (i.e., $\Delta d_c < 0$) will be underestimated with the ratio of $1-\alpha$ and the amount is added to the original slack (i.e., less positive slack). On the other hand, enlarged cell delay by up-shifted V_t will be overestimated by $1+\alpha$ times larger and subtracted from the original slack (i.e., more negative slack). The main reason of such pessimism is to prune out the "potentially invalid" cases as much as possible and reduce the redundant timing updates with P&R tool in the last loop. After all invalid cases are pruned, we sort the remaining solutions in ascending order of estimated power overhead, which is calculated in the same manner as that of estimated delay change (Line 16).

c) V_t reassignment: Lastly, we try top-k solutions in S by checking the timing preservation using the P&R tool (Lines 18–24), and obtain a violation-free solution with minimum power overhead. If no violation-free solution is found in k trials, we pick the baseline combination for c_v and cells in C_n, which reassigns all V_ts to LVT to eliminate MIA violation with guaranteed timing preservation and maximum power overhead. The total number of timing check function calls with the P&R tool is bounded by $O(nk)$ where n is the number of intra-row MIA violated cells and k is a usage limit parameter of the timing check function.

3) Speedup Technique: Step 1 should completely remove all the intra-row MIA violations, which means that Algorithm 1 should be applied on every intra-row MIA violated cell. Moreover, as a V_t reassigned cell (i.e., c_{targ} or c_{offset}) lies on multiple timing paths, it is highly possible that one V_t reassignment affects the timing information of several timing paths that the other pairs of target cells and offset cells are involved in. This makes the application of Algorithm 1 into the design a sequential process, one violated cell at a time, to guarantee the timing preservation. Thus, the reduced number of timing check function calls [$O(nk)$] is still considerable in terms of runtime, as it takes tens of hours to complete the full step for a design with thousands of intra-row MIA violations.

To cope with this issue, we devise a speedup technique to further reduce the function calls while avoiding the timing path overlap problem, as summarized in Algorithm 2. After we trace through the timing critical paths for an MIA violated cell, instead of calling the timing check function for the current V_t reassignment as in Algorithm 1, we mark all the traced cells (not only the V_t reassigned cells) as "fixed" and move on to the next violated cell without the function call. For the remaining iterations, all the marked cells are excluded from the c_{targ} or c_{offset} candidate. For example, we skip the V_t reassignment for a violated cell if the violated cell itself or its neighboring cell is in the marked cell list C_{fixed}. When tracing a timing path for c_{offset}, we also exclude the cells in C_{fixed} from the c_{offset} candidate. After a full loop, a set of V_t reassignment solutions that are parallel to each other will be stored in S', which can be solved in parallel with at most k trials to find a feasible solution or apply the baseline case. Then, we reset C_{fixed}, S', and restart the loop for the remaining violated cells.

With the speedup technique, the number of function calls reduced to $O(mk)$, where m is the number of groups with timing independent cells, which is many orders of magnitude smaller than the violated cell count ($m \ll n$).

B. Step 2: Elimination of Inter-Row MIA Violations

After all intra-row MIA violations are removed from Step 1, we formulate a MILP model from [7] to resolve inter-row MIA violations. As in [7] we only allow the V_ts to be shifted down (i.e., high-speed, more power), as there is an unpredictable number of cells related to one inter-row MIA violation and it is time-consuming to consider all V_t combinations and related timing critical paths for timing preservation as we do in Step 1.

Algorithm 2 Accelerating Step 1: Grouping Cells With Non-Overlapping Timing Paths

```
    Input: $C_v$ (intra-row MIA violated cells)
 1: $S' \leftarrow \{\}$
 2: $C_{fixed} \leftarrow \{\}$
 3: for $c_v$ in $C_v$ do
 4:     $s \leftarrow$ Solution for $c_v$ with Algorithm 1 and $C_{fixed}$
 5:     if $s$ exists then
 6:         $C_c \leftarrow$ cells with reassigned $V_t$ from $s$
 7:         for $c_c$ in $C_c$ do
 8:             for $c_{path}$ in $path(c_c)$ do
 9:                 $C_{fixed} \leftarrow C_{fixed} \cup \{c_{path}\}$
10:             end for
11:         end for
12:         $S' \leftarrow S' \cup \{s\}$
13:     end if
14: end for
15: return $S'$
```

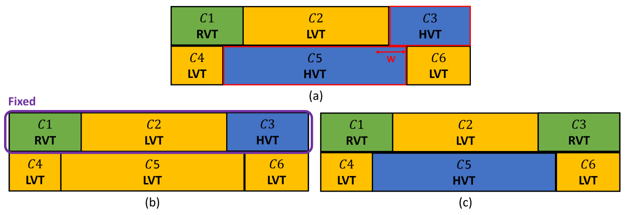

Fig. 6. Example of suboptimal solution from strip-wise MILP model. (a) Inter-row MIA violation between $C3$ and $C5$. (b) Solution with fixed top row ($C5$: HVT $\rightarrow$ LVT). (c) Solution with less power overhead ($C3$: HVT $\rightarrow$ RVT).

Instead, we add a V_t refinement step later to compensate for the power overhead (details are in Section III-C).

Our MILP model is much simpler than that of [7] with fewer constraints and variables, as we:

1) only consider inter-row MIA violations;
2) do not allow cell displacement.

For 1), we start from the design that is free from intra-row MIA violation by merging each of the narrow cells with one of their neighboring cells or minimally sized filler cells with the same V_ts, following the solution from Step 1. 2) is essential for perfect timing preservation.

Thanks to the simplified problem space, we can contain all violations into a single MILP model (i.e., without splitting into strips) and retrieve an optimal solution in a reasonable runtime. The problem size of the previous MILP model in [7] is too large to resolve all MIA violations at once. Instead, the work in [7] splits the design horizontally into multiple strips and solves them in sequence (top to bottom) while fixing the V_t assignment of the bottom-most row before solving the next strip. However, splitting the problem incurs a suboptimal solution, an example of which is shown in Fig. 6. When an inter-row MIA violation is located on the strip boundary [see Fig. 6(a)], the MILP model from the next strip will solve it by shifting V_t of $C5$ down to LVT, while there is a better solution with less power overhead by converting a smaller cell ($C3$) into a higher V_t (RVT).

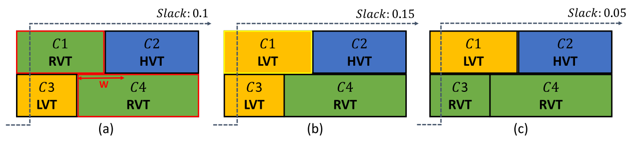

Fig. 7. Illustration of V_t refinement for power recovery. (a) Inter-row MIA violation between $C1$ and $C4$. (b) Solution from Step 2: lower V_t to $C1$. (c) V_t refinement in Step 3: higher V_t to $C3$.

C. Step 3: Recovery of Power Loss

The last step is to countervail the increased power consumption from the MILP-based solution in Step 2 that only allowed the V_t to be shifted down. It follows a similar approach as Algorithm 1. For each cell with its V_t shifted down from Step 2, we trace through its timing critical path to find candidates for offset cell, i.e., V_t of which can be raised. Among all the candidates, we prune the case if it violates MIA constraint or its estimated path slack is worse than a constraint value (i.e., the worst slack (WS) of the design), sort the remaining candidates in ascending order of estimated power overhead, and pick the first one that meets the timing constraint.

Fig. 7 shows an example. Inter-row violation between $C1$ and $C4$ [see Fig. 7(a)] is cleared by shifting down the V_t of $C1$ [see Fig. 7(b)]. It gives an extra slack to the timing path that goes through $C3$ and $C1$, which is utilized by raising V_t of $C3$ to reduce power overhead while preserving timing constraint.

IV. Preprocessing Algorithm

The main issue of our MIA violation removal algorithm in Section III is the total runtime. Despite our efforts on speeding up the algorithm, it still has a possibility of huge runtime because of its sequentiality in Steps 1 and 3 for timing preservation. As we have emphasized, we cannot compromise the timing closure because it is a critical factor in the design enablement. In this section, we introduce a different approach to reduce the runtime by removing the violations in advance and propose a preprocessing algorithm that scales down the problem space before getting into our complete removal algorithm.

As shown in Fig. 3, preprocessing is conducted at the preroute stage to reduce the number of MIA violations. It works as a buffering stage to prohibit the final layout from having a huge amount of MIA violations, which incurs an excessive runtime of our MIA violation removal algorithm.

We again use the MILP model in the preprocessing step with more different options from Section III-B, as we:

1) allow cell displacement;
2) do not allow V_t reassignment;
3) set the objective to minimize the MIA violations which are contrary to our algorithm at the final stage.

A. Allow Cell Displacement

As the preprocessing is conducted before the routing stage, we do not need to reroute any wires after cell relocation. Moreover, we can modify the design more aggressively because a little amount of timing degradation will be easily recovered in the following routing and timing closure stage. Thus, we allow cell displacement in the preprocessing with an upper bound of the total cell movement. It is expressed in the MILP model as

$$\sum_{r}\sum_{k}\bar{\delta}_{r,k} \le C_{\max} \tag{2}$$

where $\bar{\delta}_{r,k}$ is the absolute distance of kth cell in row r moved and $C_{\max}$ is the upper bound of the total cell movement.

B. Do Not Allow V_t Reassignment

In contrast, V_t reassignment is not allowed in the preprocessing for not increasing the power consumption and maintain design quality in this step. It will reduce a huge amount of the problem size of the preprocessing MILP model, as all of the variables and equations related to the V_t assignment in [7] are ignored in this model.

C. Set the Objective to Minimize the MIA Violations

The main difference of our MILP model in here from the previous model [7] is that our objective function is not to eliminate all MIA violations, but to minimize them under a certain design constraint. It is inefficient to make a MIA-violation-free design in exchange for huge design degradation at the preroute stage, since the following design stages (i.e., route and timing closure) will eventually generate some MIA violations by relocating cells or reassigning cell V_ts. Therefore, our target in the preprocessing step is to minimize the MIA violation count as much as possible with a limited amount of design modification.

In summary, the objective function of our preprocessing MILP model is set as

$$\min \sum_{r}\sum_{k} v_{r,k}^{\text{intra}} + \sum_{r}\sum_{k}\sum_{k'} v_{r,k,r+1,k'}^{\text{inter}} \tag{3}$$

where $v_{r,k}^{\text{intra}}$ and $v_{r,k,r+1,k'}^{\text{inter}}$ are binary values which are set to 1 if the kth cell in row r brings an intra-row/inter-row MIA violation and 0 otherwise, respectively. Limited design modification constraint is expressed as an equation in constraint (2). For each of the intra-row MIA violations, we add a constraint below to the MILP model

$$\begin{aligned} m_{r,k+j} - m_{r,k-1} \ge{} & (d_{r,k-1} + d_{r,k+j} - 1)W \\ & - M \cdot v_{r,k}^{\text{intra}} \\ & \forall\ j \ge 0 \text{ s.t. } \sum_{i}^{j} W_{r,k+i} < W \end{aligned} \tag{4}$$

where $m_{r,k}$ denotes an intermediate location between kth cell in a row r, $d_{r,k}$ becomes 1 when the V_t of the adjacent right cell is the same, M is a large constant. An inter-row MIA violation is expressed in the same way as (4), while $v_{r,k}^{\text{intra}}$ is changed to $v_{r,k,r+1,k'}^{\text{inter}}$.

Unlike the model in [7], our preprocessing MILP model is solved in a reasonable runtime as the size of the problem is significantly reduced by not considering V_t reassignment. After the preprocessing stage, the following design stages (routing and timing closure) relocate some cells and reassign V_ts, and

TABLE I
BENCHMARK CHARACTERISTICS AND DESIGN METRICS

Benchmarks	#Cells	#Narrow cells (%)	Clock (ns)	Density (%)	$\%V_t$ (H/R/L)	#MIA violations (vanilla)		#MIA violations (after **MILP-filler**)	
						Intra-row	Inter-row	Intra-row	Inter-row
wb_dma	2,509	23.8	0.7	74.26	24/41/35	417 (16.7%)	664 (26.5%)	62 (2.6%)	74 (2.9%)
ac97_ctrl	7,396	16.9	0.8	72.91	18/66/16	840 (11.4%)	2,125 (28.7%)	62 (0.8%)	97 (1.3%)
aes_core	10,303	30.0	1.5	75.24	13/59/28	1,869 (18.1%)	3,154 (30.6%)	398 (3.9%)	500 (4.9%)
wb_conmax	21,319	22.7	1.5	75.03	73/25/2	2,441 (11.4%)	6,921 (32.5%)	1,147 (5.4%)	1,207 (5.7%)
ldpc	52,800	37.3	2	62.20	16/37/47	14,099 (26.7%)	10,971 (20.8%)	1,438 (2.7%)	1,400 (2.7%)
ecg	114,600	37.7	1.5	71.42	34/50/16	32,003 (27.9%)	38,032 (32.2%)	4,239 (3.7%)	4,042 (3.5%)
aes_128	122,916	35.4	2	71.71	57/38/5	33,285 (27.1%)	42,741 (34.8%)	2,957 (2.4%)	5,895 (4.8%)
jpeg	268,414	25.8	2	63.39	38/42/20	46,674 (17.8%)	60,123 (22.4%)	8,252 (3.1%)	7,739 (2.9%)

generate extra MIA violations in the final layout. Still, the total violation count is decreased remarkably compare to that of the layout without preprocessing.

V. EXPERIMENTAL RESULTS

A. Experimental Setup

Our proposed MIA violation removal and the preprocessing algorithm are implemented on several benchmark circuits from OpenCore benchmark suites [13] with 28 nm commercial-grade multi-V_t standard cell library. We use Synopsys Design Compiler (DC) and ICC2 for logic synthesis and place-and-route (P&R), respectively. For our algorithm, we use ICC2 for the sign-off timing extraction in timing-aware V_t reassignment (Step 1) and V_t refinement (Step 3) and set the minimum implant width constraint (W) to eight placement sites, which involves up to 37.3% of standard cells in a design as "narrow" cells. The benchmark characteristics according to the above implementation options are listed in Table I, which includes basic design metrics, V_t ratio, and the number of MIA violations. We also show the MIA violation count after the MILP-based filler cell insertion, named as MILP-filler. It is different from our filler cell insertion method in Step 1 (see Section III-A1), in that we use a MILP model similar to that of Section IV but do not allow both V_t reassignment and cell displacement and try its best to minimize the MIA violation count only with filler insertion. The last two columns of Table I show the results, which indicate that even the best solution with maximum utilization of filler cell insertion is not enough to resolve all of the MIA violations. Results with MILP-filler work as a baseline for our experiments to verify the performance of our MIA violation removal algorithm, as well as our preprocessing step.

We compare the performance of our algorithms with the latest work [7], which is implemented by splitting the final layout of the benchmark circuits horizontally into multiple strips, formulating and solving the MILP model from top to bottom strip. As [7] provided no data on the effects of the cell displacement on the timing preservation, we conduct two different implementations based on the MILP model in [7], i.e., with and without cell displacement, and compare their results with ours. For the case with cell displacement, we set the cell displacement weight factor to 0.1 as in [7], and the nets connected to the displaced cells are rerouted afterward using engineering change order (ECO) routing in the P&R tool, which our proposed MIA violation removal algorithm does not require as it does not relocate cells. In V_t reassignment stage, parameter α is chosen according to the benchmark size. Small benchmarks require less time for timing update and thus we can try more solution candidates to check validity. Therefore, α is set to 0 for them. For the opposite reason, we set α as 0.2–0.3 for large benchmarks to choose solutions conservatively. We swept the value of k to trade-off the depth of exploration against runtime. In this work, we empirically set k to 5 for small benchmarks, and 3 for large benchmarks.

In addition, the previous work [7] introduced a simpler MILP model with reduced variables and constraints, as well as a speedup technique to divide the full design into multiple strips and solve MILP model for each strip sequentially. We apply both simplified MILP model and speedup technique when solving MILP models for previous works (i.e., with and without cell displacement) and Step 2 of our MIA violation removal algorithm. For all of the above MILP models, we use 20–50 row counts per strip values according to the benchmark size to balance between runtime and design quality (i.e., 20 for wb_dma, ac97_ctrl, aes_core, wb_conmax, and ldpc; 30 for ecg and aes_128; and 50 for jpeg). The MILP model for our preprocessing stage is neither simplified nor divided into strips, because it is already simple and solved in minutes even with all rows as one full MILP model.

B. Design Comparison

We first verify the performance of our algorithms in comparison with the previous work [7]. Table II shows the results of design quality in terms of power consumption and WS. The ratio of power is calculated by averaging the power changes with regard to the initial power consumption of each benchmark, and the ratio of WS is calculated by averaging the amount of WS changes from the initial design of each benchmark. A smaller power ratio implies that the design is power-efficient, and a larger WS ratio means the design is farther from the timing violation, i.e., more robust. Starting from the initial V_t assignment from the P&R tool that contains MIA violations, all methodologies of the previous [7] and our work have eliminated all of the violations, except for the second last column named as "Pre-proc. only" that shows intermediate design results before applying MIA violation removal algorithm. Note that the area of all designs remained unchanged after MIA removal, as we do not add or remove cells during our MIA violation removal flow. The total runtime is thoroughly compared in a separate table, which is Table III.

TABLE II

DESIGN COMPARISON BETWEEN THE PREVIOUS WORK [7] AND OUR PROPOSED ALGORITHMS FOR THE ELIMINATION OF MIA VIOLATIONS

Benchmarks	ICC2 Initial V_t assignment (Not MIA-vio.-free)		Prev. [7]				Ours					
			Cell disp. + V_t rasgn.		V_t rasgn. only		MIA vio. removal only [12]		Pre-proc. only (Not MIA-vio.-free)		Pre-proc. + MIA vio. removal	
	Power (mW)	WS (ps)	Power (mW)	WS (ps)	Power (mW)	WS (ps)	Power (mW)	WS (ps)	Power (mW)	WS (ps)	Power (mW)	WS (ps)
wb_dma	4.671	0.7	4.674	-0.4	4.694	0.5	4.676	0.6	4.637	-0.4	4.641	-0.8
ac97_ctrl	13.030	0.7	13.036	-0.8	13.063	0.7	13.052	0.8	13.004	0	13.012	0.4
aes_core	7.419	-6.2	7.477	-6.1	7.582	-0.04	7.556	-0.6	7.443	0.4	7.533	-0.5
wb_conmax	5.052	-61.6	5.125	-61.9	5.179	-62.0	5.080	-61.9	5.070	-38.8	5.115	-39
ldpc	90.516	-12.2	90.853	-31.5	91.019	-12.1	90.633	-11.8	89.807	-15.5	89.752	-32.6
ecg	68.252	-12.8	>7,200s		68.826	-12.3	68.824	-12.1	68.359	-0.6	68.733	-2.4
aes_128	50.704	-1.6	>7,200s		51.009	-3.1	50.952	-6.7	50.656	-1.3	50.788	-1.8
jpeg	144.463	-1.6	>7,200s		146.143	-1.8	146.588	-1.6	144.642	-1.1	145.980	-22.7
Ratio(Pwr) / Avg.(WS)	1.0000	0.0000	1.0054	-4.42	1.0107	+0.56	1.0070	+0.16	0.9989	+4.66	1.0038	-0.60

TABLE III

RUNTIME COMPARISON AMONG ALL DESIGN FLOWS (UNIT: SECOND)

Benchmarks	Prev. [7]		Ours			
	with disp.	w/o disp.	MIA vio. removal only [12]	Pre-proc. + MIA vio. removal		
			Step1-3	Pre-proc.	Step1-3	total
wb_dma	22	8	43	6	22	28
ac97_ctrl	22	22	112	18	78	96
aes_core	180	20	930	31	674	705
wb_conmax	103	52	944	61	655	716
ldpc	1156	685	942	71	733	804
ecg	>7,200	4,662	3,863	357	2,124	2,481
aes_128	>7,200	2,145	5,494	498	2,970	3,468
jpeg	>7,200	5,138	2,392	406	1,414	1,820
Average		0.86	1.00	0.09	0.59	0.68

TABLE IV

STEP-WISE MIA VIOLATION COUNT BREAKDOWN

Benchmarks	#MIA violations (Intra-row / Inter-row)		
	Initial	After Step 1	After Step 2
wb_dma	386 / 611	0 / 461	0 / 0
ac97_ctrl	794 / 1,974	0 / 1,583	0 / 0
aes_core	1,579 / 2,969	0 / 2,237	0 / 0
wb_conmax	2,295/ 6,163	0 / 5,147	0 / 0
ldpc	13,576 / 10,564	0 / 7,512	0 / 0
ecg	29,555 / 35,185	0 / 26,485	0 / 0
aes_128	30,230 / 39,680	0 / 33,205	0 / 0
jpeg	45,513 / 56,967	0 / 46,961	0 / 0

TABLE V

POWER CONSUMPTION CHANGES AND RUNTIME BREAKDOWN FOR OUR MIA VIOLATION REMOVAL ALGORITHM

Benchmarks	Power (mW)			Runtime (s)		
	Step1	Step2	Step3	Step1	Step2	Step3
wb_dma	4.660	4.679	4.676	31	1	11
ac97_ctrl	13.026	13.055	13.052	77	4	32
aes_core	7.438	7.584	7.556	542	6	382
wb_conmax	5.045	5.120	5.080	519	10	415
ldpc	90.503	91.077	90.633	343	24	575
ecg	68.510	69.032	68.824	1,296	69	2,498
aes_128	50.821	51.076	50.952	1,609	110	3,775
jpeg	145.302	146.611	146.588	712	282	1,398
Ratio	1.0013	1.0099	1.0070	0.485	0.024	0.491

Prior to the detailed analysis, Table IV shows the changes of the MIA violation counts during our MIA violation removal algorithm.[1] The table indicates that all intra-row MIA violations are removed after Step 1, and the remaining inter-row MIA violations are eliminated after Step 2. This MIA-violation-free status reaches out to the end as Step 3 does not generate extra MIA violation. Inter-row MIA violation count after Step 1 is slightly decreased as Step 1 changes small cell V_ts into their neighboring V_ts, which removes the related inter-row MIA violations simultaneously.

From the results between two different implementations of previous work (i.e., with/without cell displacement), we find that even a small amount of cell displacement incurs more negative slacks as it requires a complete rerouting for all nets linked to the displaced cells. Moreover, a runtime issue arises when allowing both cell displacement and V_t reassignment for the benchmark designs with more than 100k cells, which is a common scale in recent days. The results from our MIA violation removal algorithm applied at the final stage show better timing closure compared to the previous work with cell displacement, and lower power consumption compared to the previous work without cell displacement. In detail, Table V shows the power consumption changes and runtime breakdown in each step of our MIA violation removal algorithm. Our algorithm achieves a minimum power overhead with Step 1, which is mainly due to the cells that V_t are moved upward under dedicated timing-aware V_t reassignment. Then, the unidirectional V_t reassignment in Step 2 increases the power consumption, which is compensated by the V_t refinement in Step 3. All designs with our algorithm are finished much earlier than the runtime limit (2 h), of which Step 1 and Step 3 take the most. Note that the power and WS improvements are averaged and thus seem relatively small.

We also present the design results with a full flow with our preprocessing algorithm added at the preroute stage in Table II. Please note that the design results in the second last column of Table II ("Pre-proc. only") are obtained by our preprocessing stage and the subsequent routing, but before performing our MIA violation removal algorithm. These designs are thus not MIA-violation-free and represent an intermediate state to analyze the effects of our preprocessing stage. It is observed that the application of preprocessing algorithm has little or no effects on the design quality, and the final results, i.e., with preprocessing algorithm followed by MIA violation removal flow, show less power consumption compared to both implementations of the previous work. Regarding the WS change, our MIA violation removal flow makes the average WS change slightly negative. While it is possible when our pessimistic assumption fails to cover some timing critical paths in extreme cases, its amount is relatively small (less than 1 ps) and much better than that of the previous work.

[1]The initial MIA violation counts in Table IV come from the designs after preprocessing, which are different from those in Table I that come from the vanilla design, i.e., without preprocessing.

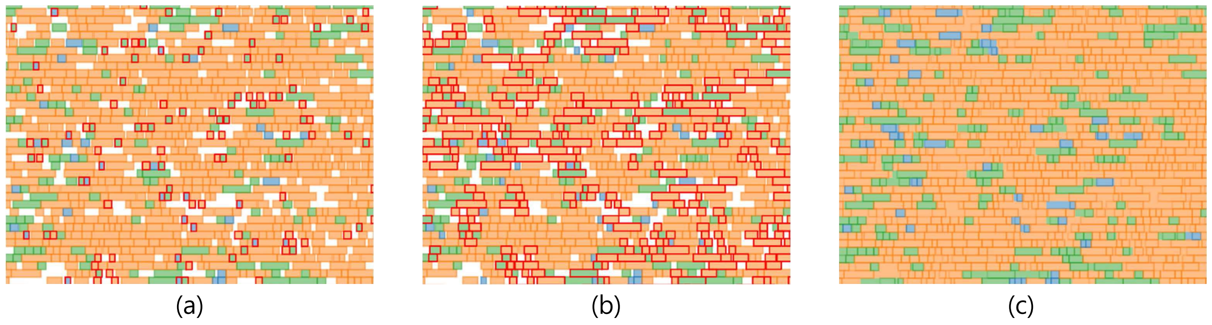

Fig. 8. Illustration of MIA violation removal in `wb_dma` benchmark circuit. Cell colors represent their different V_ts (LVT/RVT/HVT) and MIA violations are highlighted with red boxes. (a) Distribution of intra-row MIA violations. (b) Distribution of inter-row MIA violations. (c) Violation-free result after applying our algorithm.

TABLE VI
EFFECTS OF ALLOWING DIFFERENT DESIGN MODIFICATION OPTIONS IN THE PREPROCESSING ALGORITHM

	Both	V_t reassignment	Cell displacement
Power (normalized)	1.02	1.02	1.00
Runtime (normalized)	1.00	0.15	0.10
Removed #MIA vio. (%)	47.44	34.95	38.60

TABLE VII
EFFECTS OF APPLICATION POINTS OF THE PREPROCESSING ALGORITHM

	Both	Before route	Before timing closure
Power (normalized)	1.00	1.00	1.00
Runtime (normalized)	1.00	0.57	0.52
Removed #MIA vio. (%)	38.14	35.45	36.13
Rerouting?	Yes	No	Yes

As for the runtime comparison (see Table III), we include the runtime for solving MILP model and the routing time as the total runtime of the previous works, while our total runtime consists of the computing time for path & timing extraction (Synopsys ICC2), timing update (Synopsys ICC2), solving algorithms in Steps 1 and 3 (Python), and solving MILP models for Step 2 and preprocessing stage (Python). In the previous work, the MILP model with both cell displacement and V_t reassignment makes the problem too large to be finished within 2 h for some large benchmarks. Compared to the previous work without cell displacement, our MIA violation removal algorithm alone takes 16% longer runtime. Although we reduced a huge amount of the runtime for our MILP model in Step 2 by excluding cell displacement, the iterative processes in Steps 1 and 3 make the full algorithm longer, which occupies 97.6% of the total runtime. While these steps are incremental and thus the runtime can be reduced by lowering the number of trials (e.g., k) and sacrificing the design quality, we instead insert a preprocessing algorithm to reduce the total problem space and the total runtime as well. It reduces the runtime of our MIA violation removal algorithm by 41%, thereby reducing the total runtime by 32%, which is even shorter than the previous work without cell displacement by 21%. Fig. 8 shows an example of MIA violation removal with our algorithm. Both intra-row [see Fig. 8(a)] and inter-row [see Fig. 8(b)] violations are cleared in the final layout [see Fig. 8(c)].

C. Studies on the Preprocessing Algorithm

In this section, we explore in detail the various aspects of our preprocessing algorithm.

1) Design Modification: As the preprocessing algorithm will be applied in the early design stages, we can allow both V_t reassignment and cell displacement with less consideration about the design enablement. However, adding variables and constraints for both techniques in the MILP model incurs excessive runtime, as proved in Sections IV-B and V-B (e.g., Table II). We explore all different options of design modifications in the preprocessing algorithm, and the results with benchmark circuits are shown in Table VI. We achieve the largest reduction of MIA violations when we allow both techniques, while it runs for up to 10× longer than using one of the techniques. Since the total runtime is a dominant issue, we choose the option of only allowing cell displacement in the preprocessing algorithm as it shows better results than only adopting V_t reassignment in every metric (i.e., power, runtime, and reduction of MIA violations).

2) Application Point: The preprocessing algorithm is a modularized process and thus can be applied in any of the design stages (e.g., after placement, after CTS, after route, etc.). We choose two feasible candidate points among them, which are before route and before timing closure, as the preprocessing is more efficient when adopted in the later design stage. Table VII shows the comparison. Again, applying on both stages shows a slightly more reduction of MIA violations but takes around 2× longer than applying on one stage. We thus choose only one stage point to apply our preprocessing algorithm, before route, as it does not require a rerouting process that embeds a possibility of large timing distortion.

3) Preprocessing Parameter: Another important knob is the parameter regarding the upper bound of cell displacement amount, which is $C_{\max}$ in (2) of Section IV. We tune $C_{\max}$ in proportion to the total area covered by the MILP model, and Fig. 9 shows the effect of $C_{\max}$ to the total runtime and removed MIA violations. For instance, in a MILP model that covers 20 000 placement sites, allowed displacement of 1% means the maximum total displacements of all cells in the area are 200 sites. We find that the total runtime

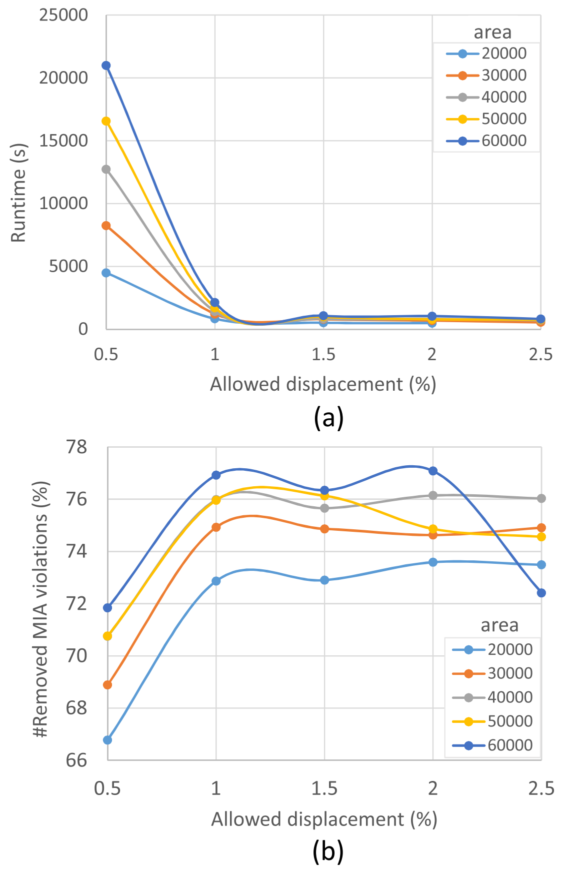

Fig. 9. Comparison among different cell displacement constraints in `aes_128` benchmark circuit. (a) Runtime of the MILP model. (b) Reduction of MIA violations.

monotonously decreases and the MIA violation reduces more as $C_{\max}$ becomes larger from 0.5% to 1% of total area, which implies that too small amount of allowed cell displacement makes it hard for the MILP model to find a feasible solution. As both metrics converge in between 1% and 2%, we find the $C_{\max}$ value for each of the benchmark circuits in this range according to the covered area of the MILP model.

4) Reduction Effects of MIA Violations: Fig. 10 verifies the performance of our preprocessing algorithm in reducing the MIA violations, which shows the MIA violation count of benchmark circuits in three different implementations and $C_{\max}$ indicates the allowed amount of cell movement percentage. $C_{\max}$ is set to 1.5% for six benchmarks (wb_dma, ac97, aes_core, wb_conmax, ldpc, and jpeg) and 2.0% for two other benchmarks (ecg and aes_128), which are specified in Fig. 10. Note that MILP-filler is applied to all designs to exclude the trivial MIA violations that can be solved by filler cell insertions without cost overhead. Compared to the initial P&R results without any MIA violation removal algorithm, our preprocessing algorithm eliminates 71% of MIA violations on average at the preroute stage. While the following stages (i.e., route and timing closure) generate some extra violations, the violation count is still 53% smaller than that of the initial P&R results on average, which provides a drastically reduced problem size for our MIA violation removal algorithm.

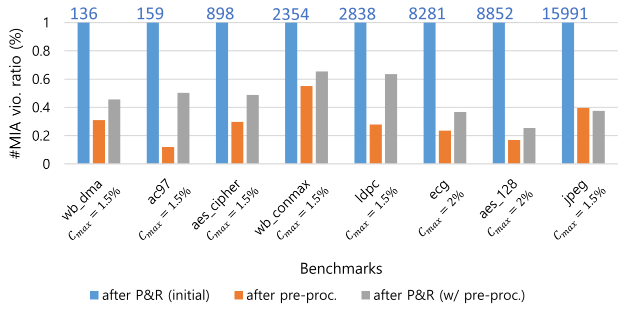

Fig. 10. Comparison of MIA violation counts (after MILP-filler).

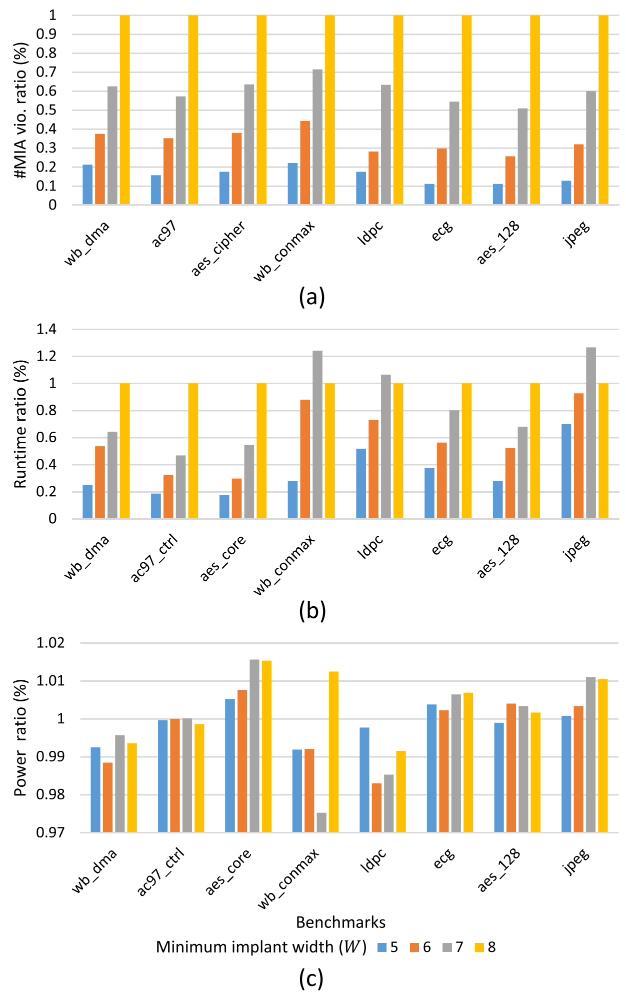

Fig. 11. Design comparison with different W values which range from 5 to 8. (a) MIA violation count. (b) Total runtime. (c) Power consumption. All values are normalized.

D. Results From Different Implant Width Constraints

Lastly, we verify our algorithms (preprocessing & MIA violation removal algorithm) on various minimum implant width values (W) from 5 to 8. Graphs in Fig. 11 compare the designs from different W values, and we successfully eliminate all MIA violations for every design. As in Fig. 11(a), the total MIA violation count is decreased down to 11% when W changes from 8 to 5, which incurs a much smaller problem space and thus shorter runtime [see Fig. 11(b)].

Fig. 11(c) compares the power consumption, which shows no tendency as in MIA violation count or runtime. This implies that our algorithms delicately tune the cell V_ts to compensate for the power loss regardless of the amount of MIA violations to be removed.

VI. Conclusion

In this article, we proposed a systematic solution to the problem of eliminating MIA violations. Unlike the conventional approaches which have placed their utmost importance on minimizing the leakage power in the course of removing MIA violations and have little or partially considered the preservation of timing constraints that are already being tightly optimized in the final stage, we placed our primary concern on meeting the timing constraints while minimizing the power overhead. Consequently, we resolved two types of MIA violations namely intra- and inter-row MIA violations step by step and took into account the timing budget in the meantime. Precisely, we proposed a three-step MIA violation removal algorithm: 1) performing bidirectional V_t reassignment to fix all intra-row MIA violations while considering timing constraints; 2) performing unidirectional (downward) V_t reassignment to fix all inter-row MIA violations while satisfying both intra-row MIA and timing constraints; and 3) recovering power loss while meeting intra- and inter-row MIA constraints as well as timing constraints. Moreover, we also introduced the preprocessing algorithm at the preroute stage to leverage more design freedom to sharply reduce the MIA violation count in advance without worsening the performance, which in turn reduced the total runtime of our MIA violation removal algorithm. Experiments with benchmark circuits showed that our approach was very effective in eliminating MIA violations with less power overhead in comparison with the existing state-of-the-art approach while not incurring timing violations. In addition, our preprocessing algorithm achieved a 53% reduction on MIA violation count beforehand and in turn reduced 32% of total runtime on average.

Acknowledgment

The Electronic Design Automation (EDA) tool was supported by the IC Design Education.

References

[1] A. Shebaita and Y. Ismail, “Multiple threshold voltage design scheme for CMOS tapered buffers,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 1, pp. 21–25, Jan. 2008.

[2] A. B. Kahng and H. Lee, “Minimum implant area-aware gate sizing and placement,” in *Proc. 24th, Ed., Great Lakes Symp. VLSI (GLSVLSI)*, 2014, pp. 57–62.

[3] K. Han, A. B. Kahng, and H. Lee, “Scalable detailed placement legalization for complex sub-14 nm constraints,” in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 867–873.

[4] G. Flach, T. Reimann, G. Posser, M. Johann, and R. Reis, “Effective method for simultaneous gate sizing and Vth assignment using Lagrangian relaxation,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 4, pp. 546–557, Apr. 2014.

[5] D. Mangiras and G. Dimitrakopoulos, “Incremental Lagrangian relaxation based discrete gate sizing and threshold voltage assignment,” in *Proc. 10th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, Jul. 2021, pp. 1–5.

[6] S.-I. Lei, W.-K. Mak, and C. Chu, “Minimum implant area-aware placement and threshold voltage refinement,” in *Proc. 21st Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2016, pp. 192–197.

[7] W.-K. Mak, W.-S. Kuo, S.-H. Zhang, S.-I. Lei, and C. Chu, “Minimum implant area-aware placement and threshold voltage refinement,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 7, pp. 1103–1112, Jul. 2017.

[8] T. Huynh-Bao et al., “Statistical timing analysis considering device and interconnect variability for BEOL requirements in the 5-nm node and beyond,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 5, pp. 1669–1680, May 2017.

[9] K.-H. Tseng, Y.-W. Chang, and C. C. C. Liu, “Minimum-implant-area-aware detailed placement with spacing constraints,” in *Proc. 53rd Annu. Design Autom. Conf.*, Jun. 2016, pp. 1–6.

[10] J. Chen, P. Yang, X. Li, W. Zhu, and Y.-W. Chang, “Mixed-cell-height placement with complex minimum-implant-area constraints,” in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2018, pp. 1–8.

[11] J. Chen, Y.-W. Chang, and Y.-Y. Wu, “Mixed-cell-height detailed placement considering complex minimum-implant-area constraints,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 10, pp. 2128–2141, Oct. 2021.

[12] E. Jeong, H. Park, and T. Kim, “A systematic removal of minimum implant area violations under timing constraint,” in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2022, pp. 933–938.

[13] *OpenCores Benchmark Suite*. Accessed: 1999. [Online]. Available: http://www.opencores.org/