A Systematic Removal of Minimum Implant Area Violations under Timing Constraint

Eunsol Jeong[1], Heechun Park[2] and Taewhan Kim[1]
[1]Department of Electrical and Computer Engineering, Seoul National University
[2]Education and Research Program for Future ICT Pioneers, Seoul National University

***Abstract*—Fixing minimum implant area (MIA) violations in the post-route layout is an essential and inevitable task for the high-performance designs employing multiple threshold voltages. Unlike the conventional approaches, which have tried to locally move cells or reassign V_t (threshold voltage) of some cells in a way to resolve the MIA violations with little or no consideration of timing constraint, our proposed approach fully and systematically 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 timing constraint and minimal power increments; (2) performing a theoretically optimal V_t reassignment to fix the inter-row MIA violations while satisfying both of the intra-row MIA and timing constraints; (3) refining V_t reassignment to further reduce the power consumption while meeting intra- and inter-row MIA constraints as well as timing constraints. Experiments through benchmark circuits show that our proposed approach is able to completely resolve MIA violations while ensuring no timing violation and achieving much less power increments over that by the conventional approaches.**

I. Introduction

Multi-V_t designs are widely employed in implementing power-aware high-performance chips [1] to reduce leakage power while meeting timing constraints. Low-V_t (LVT) standard cells drive signal faster while consuming more leakage power, and regular/high-V_t (RVT/HVT) cells operate relatively slower and less leaky. LVT cells are better to be on timing critical paths to achieve short path delay, and HVT cells are usually located on non-critical paths to save power consumption at the expense of delay increase. Designers leverage the usage of LVT/RVT/HVT cells in the context of reducing leakage power under timing constraints.

As the technology node downscales to 22nm and below, manufacturing restrictions limit the minimum area of a V_t island in the implant region [2], which is so-called *minimum implant area* (MIA) constraint. We can classify MIA violations into two types namely *intra-row* and *inter-row* MIA violations, as illustrated in Fig. 1. An instance of intra-row MIA violation, shown in Fig. 1(a), occurs when the area of a V_t island is smaller than the limit of the minimum implant area. This means, in chip implementations with row-based cell placement, the implant width should be longer than the limit, W, of minimum implant width to satisfy the intra-row MIA constraint. On the other hand, an instance of *inter-row* MIA violation [3], as illustrated in Fig. 1(b), takes place when the same V_t on adjacent rows form a staircase and its abutting width is shorter than the limit value W.

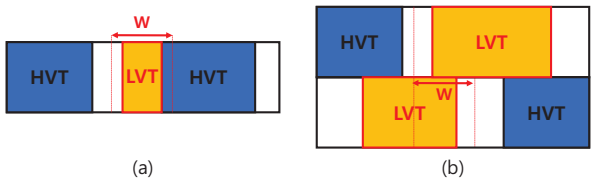

Fig. 1. Example of two types of MIA violation in row based cell placement. (a) Intra-row MIA violation. (b) Inter-row MIA violation.

Traditionally, MIA violations were considered of no importance and simply fixed by inserting filler cells of the same V_t as that of MIA violating cells. However, as the node downscales to 22nm and below, the number of occurrences of intra-row MIA violations have sharply increased, and it is now almost impossible to resolve the violations by solely relying on the filler cell insertion without increasing the die size. Furthermore, the emergence of inter-row MIA violations on advanced technology below 10nm has made resolving or avoiding both inter- and intra-row MIA violations a non-trivial task. Thus, the conventional approaches have tried to locally move cells or reassign V_t (threshold voltage) of some cells in a way to resolve the MIA violations. However, one critical point commonly missed by their approaches is little or no consideration of timing constraints. To resolve the timing violations, displacing cells or inserting repeaters is required, which again may bring up new spots with MIA violations.

In this work, we propose a systematic approach that is able to fully control 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 of the intra-row MIA and timing constraints; (3) refining V_t reassignment to further reduce the power consumption while meeting intra- and inter-row MIA constraints as well as timing constraints. (It should be noted that our approach does not exploit cell movement as it is fatal to the timing preservation in post-route layouts.)

978-3-9819263-6-1/DATE22/©2022 EDAA

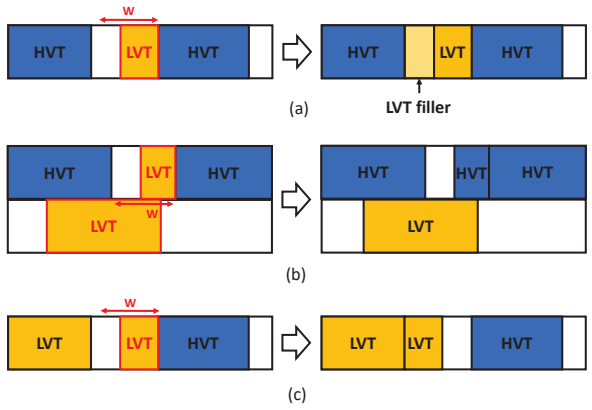

Fig. 2. Examples of fixing MIA violation through (a) filler cell insertion, (b) V_t reassignment, and (c) cell shifting.

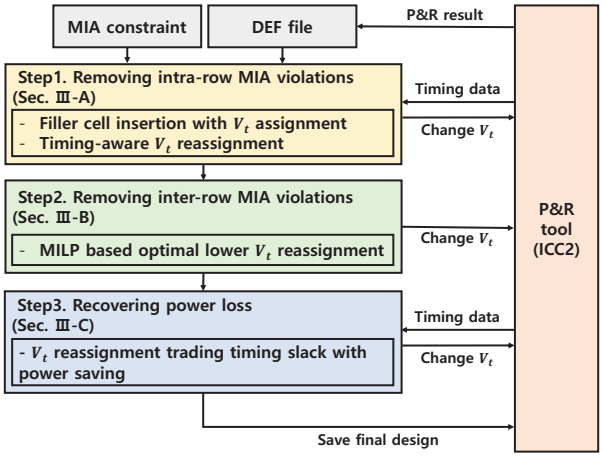

Fig. 3. Block diagram of our proposed algorithm for MIA violation removal

II. Related work

Filler cell insertion (e.g., Fig. 2(a)) is a common solution to fixing MIA violations. As V_t of a filler cell does not incur any design cost, it is possible to add filler cells with appropriate V_t to whitespaces to expand the V_t island width beyond W and solve intra-row MIA violations with no loss of design quality. Synopsys IC Compiler II (ICC2) basically provides filler cell insertion flow according to the V_t of both sides of the target whitespace and user-defined rules.

As the technology node is scaled down, filler cell insertion became no longer enough to eliminate all MIA violations and various solutions have been proposed in the literature. At the post-route, Kahng and Lee [2] proposed a heuristic algorithm of row-based V_t reassignment (e.g., Fig. 2(b)) and cell movement (e.g., Fig. 2(c)) to remove MIA violations by rearranging V_t islands to increase implant area. Later, Mak *et al.* [4] formulated the problem as mixed integer-linear programming (MILP) model to find an optimal solution. However, those solutions have not considered timing preservation and have not manipulated power loss effectively. They claimed that timing constraints will be satisfied if the baseline design (i.e., before applying the algorithms) meets timing and the cells' V_ts are only allowed to be reassigned to lower V_ts (i.e., become faster). Nevertheless, timing preservation is also affected by cell movement as it is required to re-route the wires connected to the displaced cells. Since the impact of cell movement and re-routing on the path delay becomes significant in sub-10nm technology [5], even a minimized cell displacement as in [4] 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.

At the placement, Tseng *et al.* [6] proposed a cluster-based detailed placement flow to cluster cells with identical V_t. Han *et al.* [3] proposed an MILP-based algorithm to fix all MIA constraints in the detailed placement. Chen *et al.* [7] proposed a two-stage method to resolve the MIA problem in mixed-cell-height-based design at the global placement and legalization stage. All these solutions, however, are not practical in that they do not consider the effects of inserting or displacing cells on the post-placement optimizations i.e., pre-CTS optimization, CTS, post-CTS optimization, and post-route optimization.

III. Algorithm for Removing MIA Violations

Fig. 3 shows the block diagram of our proposed algorithm for removing MIA violation at the post-route stage. It consists of three steps. Step 1 is to eliminate all intra-row MIA violations by heuristically performing V_t reassignment while preserving the timing constraint as well as minimizing the amount of the power increase. Step 2 is then 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 in a way to find a set of cells to be lower V_t reassignment of the least power increase. Step 3 is to recover the power loss caused by the lower V_t reassignment in Step 2 by iteratively performing V_t reassignment to some cells on the less critical paths induced in Step 2 while not violating the intra- and inter-row MIA constraints as well as timing constraints met in Step 2.

A. Step 1: Elimination of intra-row MIA violations

This step performs as a preprocessing step V_t assignment to the filler cells, followed by applying a timing-aware V_t reassignment. In addition, we provide a speedup technique to accelerate the process of the V_t reassignment.

1) Filler cell insertion with V_t assignment: As a preprocessing step, we apply a filler cell insertion if it leads to removing an intra-row MIA violation on the small cell(s) adjacent to the filler cell. This step reduces the problem complexity to be performed in the main steps. Precisely, there are two cases of intra-row MIA violations that are treated by filler cell insertion: When a narrow cell (i.e., width $< W$) has enough whitespaces for filler cell to solve its violation (e.g.,

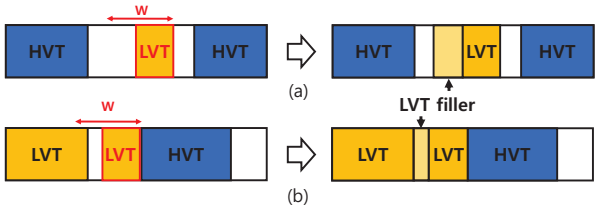

Fig. 4. Cases of filler cell insertion with V_t assignment

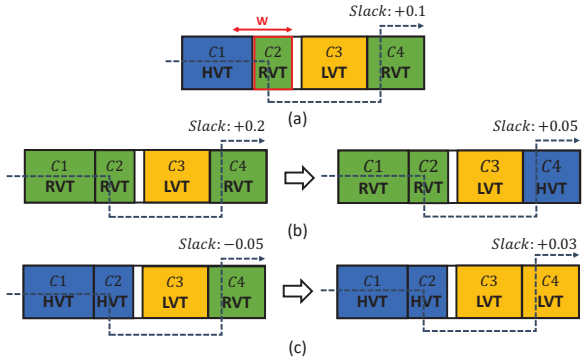

Fig. 5. Illustration of intra-row MIA violation removal. (a) Intra-row MIA violated cell ($C2$) without enough whitespaces. (b) Solution with lower V_t to $C1$. (c) Solution with higher V_t to $C2$.

Fig. 4(a)); and when a narrow whitespace is located between cells with the same V_t (e.g., Fig. 4(b)). Note that we add filler cells to a minimum amount and leave whitespaces as much as possible in this step, and the remaining whitespaces will be fully utilized in our inter-row MIA violation removal in Step 2 in Sec. III-B. Note that this step never affects timing.

2) Timing-aware V_t reassignment: We resolve intra-row MIA violations that are not able to be fixed with filler cell insertion. We apply V_t reassignment to a violated cell or a neighboring cell to form a larger V_t island and avoid violation. In the meantime, we also trace the timing critical path of the V_t reassigned cell (say *target cell*) or its neighboring 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 [2], [4] 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 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) to assign RVT cell to $C1$ (Fig. 5(b)) or (2) to assign HVT cell to $C2$ (Fig. 5(c)). 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: **if** s incurs violation or $slack < slack_{worst}$ **then**
13: $S \leftarrow S - \{s\}$
14: **end if**
15: **end for**
16: **Sort** S in ascending order of estimated power overhead
17: **for** s in S **do** ▷ V_t reassignment
18: Update V_t according to s
19: **if** slack of $path(c_{targ}) < 0$ **then**
20: **return**
21: **end if**
22: **end for**

case 1, $C1$ is the target cell and its V_t became lower (HVT $\rightarrow$ RVT) to remove violation from $C2$. This V_t reassignment also increases the positive slack of timing critical path through $C1$, from 0.1 to 0.2. Then, we continue to perform another step to trace the timing critical path of $C1$, and find an offset cell candidate (e.g., $C4$) that is possible to have higher V_t without raising another intra-row MIA violation and preserve timing (i.e., slack > 0) as well. For case 2, the target cell is $C2$, which is assigned to a higher V_t (RVT $\rightarrow$ HVT). This incurs the decrease on the timing critical path through $C2$, which became negative and violated 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.

Algorithm 1 summarizes our timing-aware V_t reassignment algorithm for an intra-row MIA violating cell c_v, which is operated through all violating cells to find a solution that all inter-row MIA violations are eliminated.

Collect timing data: For an intra-row MIA violated cell (c_v), we firstly find violation-free V_t combinations of 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, if its V_t can be shifted in the opposite direction of that of c_{targ}. We then collect the timing data (i.e., delay and current slack) for the offset cell candidates and store the V_t combination in the queue of solutions (Line 4-12).

Prune invalid solutions: From the collected solutions, we

prune cases that are obviously invalid, which incur extra intra-row MIA violation by c_{offset} or the estimated path slack after V_t reassignment is worse than a constraint value ($slack_{worst}$) (Line 13-20). Estimated path slack is calculated based on a proportional increase/decrease of c_{targ} and c_{offset} delays in T, which is not an actual value. Still, this step is essential to reduce the runtime of the algorithm, and please note that a feasible solution always exists, which is the case to reassign all V_ts of c_v and C_n to the lower V_t and satisfies MIA constraint at the same time (we call it *baseline combination*). After that, we sort the remaining solutions in ascending order of estimated power overhead, which is calculated in the same manner as that of path slack (Line 21).

V_t reassignment: Then, we apply V_t reassignments in each solution and check the timing preservation through the P&R tool. If it cannot satisfy timing constraint (i.e., negative slack), we try the next solution and continue until it meets timing (Line 22-29). At last, a violation-free solution with minimum power overhead will be applied. The total number of calls for timing checks in Algorithm 1 is bounded by $O(n^k)$ where n is the violated cell count and k is a limited trial count for the last loop in Algorithm 1 since at most k timing checks should be performed to find a V_t combination for each of the n violated cells. (After k trials, we pick the baseline combination and move to the next violated cell.)

3) Speedup technique: Algorithm 1 should be applied on every intra-row MIA violated cell to completely remove the violation. As at least one timing check command (i.e., **update_timing** in ICC2) is conducted for each cell, the total runtime becomes considerable if we perform the algorithm sequentially. Moreover, it is dangerous to solve all violated cells in one shot or treat some cells in parallel, as the timing critical paths of target cells may overlap, and thus V_t reassignment of one cell affects the timing information of the other cases. Consequently, the timing change from the current violated cell should be updated onto the design before we move on to the next violated cells with overlapped timing paths.

We devise a speedup technique to reduce the number of function calls for timing check while avoiding timing overlapping issues, as summarized in Algorithm 2. It helps to find a set of solutions that are timing independent. Specifically, when we trace through timing critical paths during the execution of Algorithm 1, we mark the traced cells as 'fixed', so that they will be excluded from the candidates of c_{targ} or c_{offset} when treating the following violated cells. We skip calling the timing check function until the last violated cell is treated, and use the call for all V_t changes from S'. Then, we reset C_{fixed} and iterate the flow for the rest of the violated cells. By employing this speeding up technique, the number of calls for timing checks can be reduced to $O(m*k)$ where m is the number of groups with timing independent cells ($m << n$) since the k trials for timing check can be performed on all cell elements in a group in parallel.

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** ▷ Mark all cells in path of C_c
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'

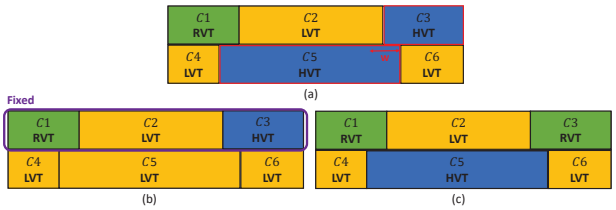

Fig. 6. An example of sub-optimal solution from strip-wise MILP model. (a) Inter-row MIA violation between $C3$ and $C5$. (b) Solution with fixed top row ($C5$: HVT→LVT). (c) Solution with less power overhead ($C3$: HVT→RVT).

B. Step 2: Elimination of inter-row MIA violations

After all intra-row MIA violations are removed beforehand, the MILP model from [4] is formulated to resolve inter-row MIA violations. Here 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. Instead, we add a V_t refinement step later to compensate for the power overhead (details are in Sec. III-C). Our MILP model is originated from [4], but is much simpler with fewer constraints and variables, as we:

- (i) only consider inter-row MIA violations;
- (ii) do not allow cell displacement.

For (i), we merged narrow cells with one of their neighboring cells or minimally sized filler cells with the same V_ts. (ii) is essential for perfect timing preservation.

As the size of the originally proposed MILP model for a problem in [4] is too large to resolve all MIA violations at once, the design is split horizontally into multiple strips. It is guaranteed that there is no inter-row MIA violation between strips by processing the strip in the order of top to bottom and fix the V_t assignment of the bottom-most row of the last strip.

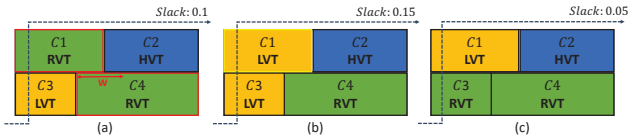

Fig. 7. Illustration of V_t refinement for power recovery. (a) Inter-row MIA violation between $C1$ and $C4$. (b) Solution from MILP model: lower V_t to $C1$ (c) V_t refinement: higher V_t to $C3$

However, splitting the problem incur a sub-optimal solution, as shown in Fig. 6. When an inter-row MIA violation is located on the strip boundary (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 an obvious solution with less power overhead by converting a smaller cell ($C3$) into a higher V_t (RVT). Thanks to the simplified problem space, we are able to contain all violations into a single MILP model and retrieve optimal solutions in a reasonable runtime.

C. Step 3: Recovery of power loss

The last step is to countervail the increased power consumption from the MILP-based solution 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 by the MILP model, 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, 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 $C4$ (Fig. 7(a)) is cleared by shifting down the V_t of $C1$ (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. Experimental Results

We tested our proposed MIA violation removal procedure on several benchmark circuits taken from OpenCore benchmark suites [8], which were implemented with multi-V_t standard cell library of 28nm foundry technology. We used Synopsys Design Compiler (DC) and Synopsys IC Compiler II ($ICC2$) for logic synthesis and place-and-route (P&R), respectively, and we also used $ICC2$ for the sign-off level timing extraction in timing-aware V_t reassignment of Step 1 and V_t refinement of Step 3. Minimum implant width (W) is set to 8 placement site steps, which involves up to 37.3% of standard cells in a design as 'narrow' cells. The benchmark characteristics are listed in Table I, which includes basic design metrics, the number of MIA violations, and V_t ratio.

To implement the latest state-of-the-art work [4] for comparison, we formulate the MILP model from the post-route layout of the benchmark circuits. According to the speedup technique used in [4], we split the design horizontally into multiple strips and apply the MILP model to one strip at a time from top to bottom. Since [4] provided no data on how badly the cell displacement affects timing preservation, we conduct two different styles of implementation based on the MILP model in [4], i.e., with and without cell displacement, and compare their results with ours. With cell displacement, it is required that the nets connecting displaced cells should be rerouted afterward through engineering change order (ECO) routing. In contrast, note that our proposed MIA removal method never allows cell movement.

Table II shows a comparison of the leakage power and timing in terms of the value of the worst slack (WS) extracted by using a commercial P&R tool ($ICC2$). All methodologies have completely eliminated all MIA violations for all benchmark circuits. From two sets of results produced by the existing work in [4], we observe that allowing cell displacement worsens the timing closure for most testcases, though the power consumption becomes slightly lower. We set a weighting factor of 0.1 to the cell displacement as [4] does. Thus, the results with cell displacement do not show a drastic increase of worst negative slack since shifting down V_ts cancels some of the effects. Nevertheless, since the MIA violation removal is applied after all physical design flow is passed, even a small deterioration in timing closure tends to cause a huge negative effect on productivity. The comparison of power and timing in Table II shows that our proposed method outperforms both options of the prior method. The table also reports the power consumption changes for the designs optimized by the sequential application of the three steps in our method. It shows that our method achieves the largest power saving after Step 1, which is mainly due to the raising of V_ts with dedicated timing-aware V_t reassignment. Power consumption is then increased after Step 2, which only allows uni-directional V_t changes (i.e., downwards), and it is compensated by the V_t refinement at Step 3.

Lastly, Table III reports runtime comparison. It shows that allowing cell displacements increases the problem size and corresponding MILP formulations, which in turn requires more runtime to solve the model. In our method, a simplified MILP model formulation and its execution are conducted only in Step 2, where the runtime is negligible in comparison with that in [4] as well as other steps in our proposed flow. All equations and variables regarding intra-row MIA violations and cell displacements are excluded from our MILP formulation, which brings a huge reduction in the MILP model size. Thus, we are able to spend more time on Steps 1 and 3, which perform sign-

TABLE I
Benchmark characteristics and design metrics for baseline implementation

	#Cells	#Narrow cells (%)	Clock (ns)	Util. (%)	#Intra-row MIA vio.	#Inter-row MIA vio.	%V_t (H/R/L)
wb_dma	2,509	23.8	0.7	74.26	417 (16.7%)	664 (26.5%)	24/41/35
ac97_ctrl	7,396	16.9	0.8	72.91	840 (11.4%)	2,125 (28.7%)	18/66/16
aes_core	10,303	30	1.5	75.24	1,869 (18.1%)	3,154 (30.6%)	13/59/28
wb_conmax	21,319	22.7	1.5	75.03	2,441 (11.4%)	6,921 (32.5%)	73/25/2
ldpc	52,800	37.3	2	62.20	14,099 (26.7%)	10,971 (20.8%)	16/37/47
ecg	114,600	37.7	1.5	71.42	32,003 (27.9%)	38,032 (32.2%)	34/50/16
aes_128	122,916	35.4	2	71.71	33,285 (27.1%)	42,741 (34.8%)	57/38/5

TABLE II
COMPARISON OF THE TIMING AND LEAKAGE POWER OF THE DESIGNS PRODUCED BY THE STATE-OF-THE-ART EXISTING METHOD [4] AND OUR PROPOSED METHOD FOR REMOVING MIA VIOLATIONS

Benchmarks	ICC2 (initial V_t assignment)		[4] (cell move + V_t reassignment)		[4] (V_t reassignment only)		Ours (V_t reassignment only)			
	Power (mW)	WS (ps)	Power (mW)	WS (ps)	Power (mW)	WS (ps)	Power (mW)			WS (ps)
							Step 1	Step 2	Step 3	
wb_dma	4.671	0.7	4.674	-0.4	4.694	0.5	4.660	4.679	4.676	0.6
ac97_ctrl	13.030	0.7	13.036	-0.8	13.063	0.7	13.026	13.055	13.052	0.8
aes_core	7.419	-6.2	7.477	-6.1	7.582	-0.04	7.438	7.584	7.556	-0.6
wb_conmax	5.052	-61.6	5.125	-61.9	5.179	-62.0	5.045	5.120	5.080	-61.9
ldpc	90.516	-12.2	90.853	-31.5	91.019	-12.1	90.503	91.077	90.633	-11.8
ecg	68.252	-12.8	>7200s		68.826	-12.3	68.510	69.032	68.824	-12.1
aes_128	50.704	-1.6	>7200s		51.009	-3.1	50.821	51.076	50.952	-6.7
Ratio	1.0000		1.0054		1.0106		1.0006	1.0092	1.0059	

TABLE III
RUNTIME COMPARISON (UNIT: SECOND)

Benchmarks	[4] with displacement	[4] w/o displacement	Ours			
			Step1	Step2	Step3	total
wb_dma	22	8	31	0.7	11	43
ac97_ctrl	22	22	77	4	32	112
aes_core	180	20	542	6	382	930
wb_conmax	103	52	519	10	415	944
ldpc	1156	685	343	24	575	942
ecg	>7200	4662	1296	69	2498	3863
aes_128	>7200	2145	1609	110	3775	5494

off level timing analysis with a commercial CAD tool ($ICC2$). Note that Steps 1 and 3 can be applied incrementally, which means that their efficiencies depend on the time allowed for their executions. The more time we spend on Steps 1 and 3, the more power saving we could have from more dedicated timing analysis to find a better offset cell and raise its V_t without timing violations. To leverage between runtime and power efficiency, we set and tune the runtime limitation value for timing-aware V_t assignment, and tune this value to increase runtime for a more power-efficient value or vice versa.

V. CONCLUSION

In this paper, we proposed a systematic solution to the problem of completely eliminating MIA violations exposed in the post-route layouts. 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, already being tightly optimized in the post-route stage, we placed our primary concern on meeting the timing constraints. Consequently, we resolved two types of MIA violations namely intra- and inter-row MIA violations step by step while taking into account the timing budget all the time. Precisely, we proposed a three-step MIA removal algorithm: (1) performing downward/upward V_t reassignment to fix all intra-row MIA violations while considering timing constraints; (2) performing downward V_t reassignment to fix all inter-row MIA violations while satisfying both intra-row MIA and timing constraints; (3) recovering power loss while meeting intra- and inter-row MIA constraints as well as timing constraints. Through 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.

Acknowledgement: This work was supported in part by Samsung Electronics Company, Ltd. under IO201216-08205-01, FOUNDRY-202010DD003F, and EDA cluster project, in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MEST) under 2020R1A4A4079177 and 2021R1A2C2008864, in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2021R1I1A1A01049399), in part by the Institute of Information and communications Technology Planning and Evaluation (IITP) grant funded by Korea government (MSIT) under 2021-0-00754, Software Systems for AI Semiconductor Design), in part by National R&D Program through NRF funded by Ministry of Science and ICT under 2020M3H2A1078119, and in part by the BK21 Four Program of the Education and Research Program for Future ICT Pioneers, Seoul National University in 2021. The 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," *Transactions on Circuits and Systems II: Express Briefs*, vol. 55, no. 1, pp. 21–25, 2008.

[2] A. B. Kahng and H. Lee, "Minimum Implant Area-Aware Gate Sizing and Placement," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2014, p. 57–62.

[3] K. Han, A. B. Kahng, and H. Lee, "Scalable detailed placement legalization for complex sub-14nm constraints," in *International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 867–873.

[4] W.-K. Mak, W.-S. Kuo, S.-H. Zhang, S.-I. Lei, and C. Chu, "Minimum Implant Area-Aware Placement and Threshold Voltage Refinement," *Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 36, no. 7, pp. 1103–1112, 2017.

[5] T. Huynh-Bao, J. Ryckaert, Z. Tőkei, A. Mercha, D. Verkest, A. V.-Y. Thean, and P. Wambacq, "Statistical Timing Analysis Considering Device and Interconnect Variability for BEOL Requirements in the 5-nm Node and Beyond," *Transactions on Very Large Scale Integration Systems (TVLSI)*, vol. 25, no. 5, pp. 1669–1680, 2017.

[6] K.-H. Tseng, Y.-W. Chang, and C. C. C. Liu, "Minimum-implant-area-aware detailed placement with spacing constraints," in *Design Automation Conference (DAC)*, 2016, pp. 1–6.

[7] J. Chen, P. Yang, X. Li, W. Zhu, and Y.-W. Chang, "Mixed-Cell-Height Placement with Complex Minimum-Implant-Area Constraints," in *International Conference on Computer-Aided Design (ICCAD)*, 2018, pp. 1–8.

[8] OpenCores benchmark suite. http://www.opencores.org/.