SLO-ECO: Single-Line-Open Aware ECO Detailed Placement and Detailed Routing Co-Optimization

*Joong-Won Jeon, †‡Andrew B. Kahng, *Jae-Hyun Kang, *Jaehwan Kim, and ‡Mingyu Woo
†CSE and ‡ECE Departments, UC San Diego, La Jolla, CA, USA.
*Samsung Electronics Co., Ltd., Hwaseong-si, Gyeonggi-do, South Korea.
{abk, mwoo}@ucsd.edu, {joongwon.jeon, jh0717.kang, hwany.kim}@samsung.com

Abstract*—Reducing design rule check (DRC) violations, subject to meeting performance, area and schedule targets, has always been the key metric for VLSI physical design. In leading-edge technology processes under 7nm, a new *pattern-specific* form of *single-line-open* (SLO) DRC violations [11] must be avoided in the final-routed GDS layout. In this work, we propose a new methodology and open-source framework, *SLO-ECO*, that 1) reduces DRC violations beyond where commercial P&R tools saturate, and 2) actively mitigates SLO violations by performing simultaneous detailed placement and routing optimizations in small switchboxes. Our methodology extracts switchboxes from the entire layout, focusing on SLO and DRC hotspots. These switchboxes are then translated into placement and routing grids for application of a satisfiability modulo theories (SMT) solver to find DRC-free layout solutions. To track the direction of routed metal segments, we utilize *OpenDB*'s *dbWireGraph* [18] to sequence the metal segments and generate the initial and ending metal segments at the switchbox boundary. Our pin generation flow, using *dbWireGraph*'s encoding and decoding APIs, reduces runtime by over 100× compared to the previous work, CoRe-ECO [4]. We also apply multi-threading based on each SMT switchbox trial. Our experimental studies show that SLO-ECO achieves average wirelength reduction of 0.368%, along with average decrease in both DRC and SLO violations of 45.14%, within an average runtime of 15.64 hours (fully automated) across a suite of open-source benchmarks with between 11K and 70K instances.

I. Introduction

IC physical implementation is well-known to face challenges of NP-hard subproblems (e.g., placement and routing), along with exploding complexity of design sizes and layout design rules in advanced nodes. In this context, final place-and-route (P&R) applies numerous heuristics to deliver satisfactory results within turnaround time constraints. A key performance indicator of the P&R tool is the number of design rule check (DRC) violations seen at the end of final routing. These last violations are presumed to be beyond the P&R tool's ability to fix, and require efforts from human-expert physical design engineers who can fix ∼100-300 DRCs per day.

The advent of extreme ultraviolet (EUV) lithography has enabled much smaller feature sizes and greater circuit density compared to previous deep-ultraviolet immersion lithography. However, as applied in advanced sub-7nm foundry nodes, EUV has introduced *stochastic-induced* defects due to various factors, including quantum effects and photon shot noise. As a result, EUV stochastic defects consist of 1) microbridges, 2) broken lines (also known as *single-line-open*, or SLO), and 3) missing or merging contacts [1]. Figure 1 shows two examples of stochastic-induced defects in which single-line-open (SLO) violations occur. SLO defects are understood to arise from *specific routed patterns* in sub-7nm processes, and potentially lead to malfunctions in the final IC product. We note that scaling boosters such as buried power rails and backside power delivery lead to greater track utilization in minimum-pitch metal layers, which increases the risk of SLO defects.

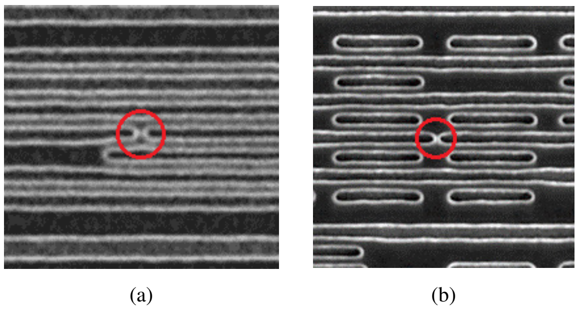


Fig. 1. The captured example showcases single-line-open (SLO) layouts in the sub-7nm manufacturing process: (a) a potential pattern, as described in [11], and (b) an alternative possible pattern. The metal segment's exposure could occur due to stochastic characteristics inherent in EUV lithography.

Current industry practice addresses stochastic-induced defects such as SLO in EUV lithography by applying *pattern matching rules* to identify all potential defects within the routed layout. Recent research [11] has proposed an automated approach to rectify risky layout patterns by widening metal, spreading metal, or expanding the via-to-metal overlap in the post-routed layout. However, such an approach is invoked *after* final routing, and requires loops back to the layout tool that can be non-convergent. If SLO violations can be mitigated in the earlier *automated* physical implementation flow (e.g., the final "engineering change order" (ECO) stages of place-and-route), this would significantly reduce design turnaround time – and/or enable additional cycles of design power, performance, area and cost (PPAC) improvement within a fixed design schedule.

In this paper, we propose a new ECO methodology, called *SLO-ECO*, that mitigates occurrence of given SLO patterns as well as traditional DRCs in advanced manufacturing nodes. Our proposed flow helps to reduce the violations left at the end of current routing tools' heuristics, and offers the potential to reduce design schedules while improving P&R outcomes. The main contributions of SLO-ECO are as follows.

- SLO-ECO provides a detailed routing methodology that

directly addresses SLO violations, formulating the avoidance of SLO-violating patterns as satisfiability modulo theories (SMT) constraints.
- We address SLO violations by considering detailed placement and routing simultaneously (in switchboxes). Our method reduces both the number of SLO violations and the number of DRC violations that are left unsolved by the commercial P&R tool.
- We propose an efficient method utilizing open-source tools (notably, OpenDB [18] and R-Tree [7]) for sorting and extracting the beginning and ending metal segments at a given switchbox boundary. Our method achieves over 100$\times$ faster runtime for switchbox generation across various designs compared to the implementation of [4].
- Starting from the final results of an existing P&R tool, our SLO-ECO implementation successfully reduces DRC and SLO violations by up to 67.46%, and improves wirelength by up to 1.5%, across the 14 benchmarks that we study.
- We implement SLO-ECO under OpenROAD [22] infrastructure and permissively open-source our code [20], thus contributing to EDA research baselines as well as the open-source EDA community.

In the following, Section II briefly reviews related works. Section III provides background from the previous *CoRe-ECO* work of [4], which we use as a starting point for SMT-based simultaneous detailed placement and routing. Section IV gives details of our proposed flow. Section V provides experimental studies that demonstrate reductions of DRC and SLO violations, and we conclude in Section VI.

II. Related Works

Substantial research and industry tool development have focused on various "last-mile" forms of engineering change orders (ECO) in chip design. While these efforts span a number of distinct design flow goals, the basic philosophy is to make local improvements without disrupting any preceding work to "close" the design.

A well-studied example of the "ECO" context is post-route leakage optimization, where threshold voltage (VT) swap and poly bias (transistor channel length) swap degrees of freedom are leveraged to improve leakage or timing without changing the final detailed routing. Lee et al. [12] introduced an ECO leakage optimization technique using graph convolutional networks (GCN) to predict VT swaps. They also proposed a VT reassignment heuristic to address minimum implant width (MIW) violations and negative slack timing impacts. Cao et al. [2] presented ECO leakage optimization frameworks utilizing graph neural networks (GNN) and bidirectional long short-term memory (BiLSTM), achieving a significant 10$\times$ runtime improvement in leakage power recovery flow compared to a commercial timer.[1]

More disruptive ECOs have also been well-studied. For gate sizing, Cheng et al. [3] proposed *DAGSizer*, a GCN-based directed acyclic graph gate sizer that uses captures timing graph structure from the netlist. Improving manufacturing yield by reducing diffusion height changes in standard-cell rows is the driving consideration for the work of Heo et al. [9], which introduced a dynamic programming-based detailed placement considering multi-height detailed placements with neighbor diffusion effects (NDE). Lin et al. [13] add minimum implant area (MIA) rule awareness to detailed placement and VT swaps for ECO leakage optimization.

The most relevant thread of research on ECO methods has addressed design closure with respect to design rule correctness, using combinatorial optimizations to find legal layout solutions (if such solutions exist). In routing optimization, Han et al. [8] recommended an integer linear programming (ILP)-based router for sub-20nm process design rule optimization. Park et al. [14] introduced an SMT-based detailed routing framework that significantly reduced runtime compared to the previous method of [8]. The emergence of the SMT-based routing framework brought further ECO research aimed at co-optimizing detailed placement and routing to tackle design rule violations. Cheng et al. [4] proposed the concurrent refinement of detailed place-and-route for ECO automation, referred to as *CoRe-ECO*. The seminal *CoRe-ECO* work introduces a placement and routing grid structure for SMT, and formulates the concurrent optimization of both detailed placement and detailed routing using SMT. While *CoRe-ECO* is able to successfully resolve all DRC violations in some designs, it still encounters limitations regarding violations from the physical verification flow. Moreover, the *CoRe-ECO* workflow reported in [4] actually includes multiple combined scripts, causing significant disk-IO overheads and inefficiencies in extracting switchboxes from around DRC hotspots. (We describe our method of addressing this inefficiency in Sections III and IV below.) In the next section, we review important aspects and background details of the *CoRe-ECO* approach.

III. Preliminaries: Key Elements of *CoRe-ECO*

The earlier work by Cheng et al. [4] introduces ECO frameworks that optimize detailed placement and routing concurrently. Our work starts by adopting most concepts proposed in their work, and indeed develops our current C++ implementation starting from $\sim$17k lines of Perl code provided by authors of [4]. In this section, we provide a brief review and explanation of the key concepts essential for the SMT formulation, aligning our discussion with the development given in [4].

A. Switchbox Structure

The switchbox plays a crucial role in SMT-based routing frameworks, enabling scalability despite the complex time complexity of the SMT optimizer. Within the post-routed layout, multiple violation *hotspots* exist, and are typically self-flagged by the P&R tool as it terminates its efforts. To address the DRC or SLO-violated hotspots, we create a three-dimensional switchbox in the hotspot region, retaining only the beginning and ending metal line segments. An example of an extracted switchbox is depicted in Figure 2. It consists of

[1]Industry methods are typically based on incremental static timing analysis, and hence have names such as *Tempus-ECO*, *PrimeTime-ECO*, *Tweaker*, etc.

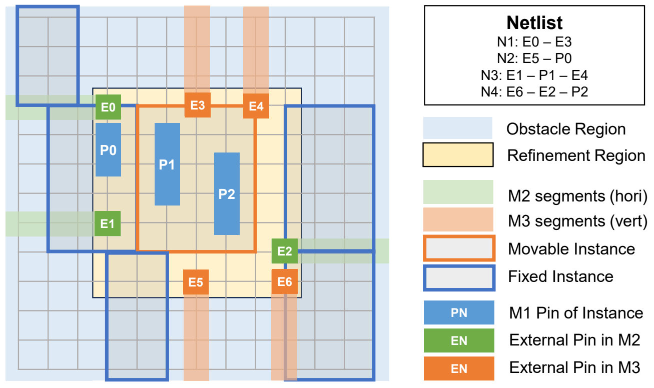


Fig. 2. An example of extracted switchbox which has width = 7 ∗ metal pitch(MP) and height = 7 ∗ MP. The ECO router will route within the refinement region (yellow).

six instances (one movable and five fixed instances), three M1 pins from instances (P0, P1, and P2), and seven external pins (E0, E1, and E2 on layer M2 and E3 through E6 on layer M3), associated with four nets (N1 to N4). The obstacle region is necessary to prevent violations of DRC rules, and additional margins beyond the refinement region serve to inform the SMT router of such violations. Subsection IV-A gives more details of the efficient generation of external pins using the open-sourced wire graph data structure.

B. Placement Grids and Relative Positioning Constraints (RPC)

The movement and orientation of each standard cell can influence pin accessibility and routability. In regions close to DRC hotspots within a routed layout, achieving a satisfying solution (for the corresponding SMT instance) becomes challenging due to the high densities of both cells and pins. To account for cell movement and orientation, [4] established placement grids for detailed placement and applied *relative positioning constraints* (RPC).

$$\begin{aligned} x_t \geq x_s + w_s \quad &\text{if} \quad x_t \geq x_s + w_s \\ x_t + w_t \leq x_s \quad &\text{if} \quad x_t + w_t \leq x_s \\ &\text{else} \quad UNSAT \end{aligned} \tag{1}$$

The RPC (1) is employed for SMT constraints to prevent overlaps between each instance. The first equation $x_t \geq x_s + w_s$ signifies that instance t must be placed to the right of instance s. The second equation $x_t + w_t \leq x_s$ denotes that instance s must be positioned to the right of instance t. If not, the SMT constraint would yield unsatisfiable (*UNSAT*), preventing illegal placements. Additionally, the boundary of each row is taken into account to prevent placements beyond the provided switchbox limits. The orientations of each instance are encoded as 0 or 1, with 1 indicating an instance flipped about the y-axis, while 0 signifies no flipping. The x-axis flipping is inherently considered based on the row locations. We systematically enumerate all potential placements and orientations within the specified switchbox.

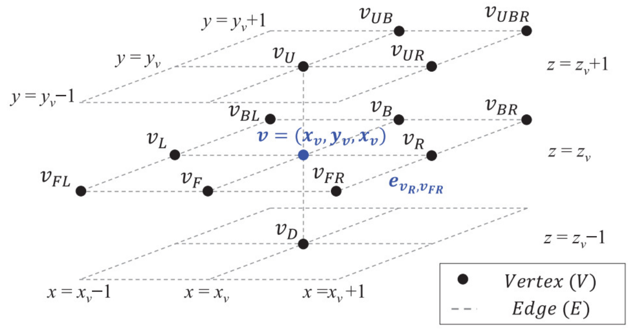


Fig. 3. Illustration of three-dimensional routing grids, from [14].

TABLE I
OBJECTIVE USED IN OUR FRAMEWORK.

Order	Variable	Definition
1	ΔV	Vertical movement of cells
2	ΔH	Horizontal movement of cells
3	ΔF	Cell flipping
4	ΔP	Pin extension
5	ML{#$VIA12$}	Number of routed segments in VIA12 layer
6	ML{#$M2$}	Number of routed segments in M2 layer
7	ML{#$VIA23$}	Number of routed segments in VIA23 layer
8	ML{#$M3$}	Number of routed segments in M3 layer

C. Routing Grids and Design Rule Constraints

The SMT routing grid, as depicted in Figure 3 (reproduced from [14]), maintains a *width/height* identical to the metal pitch of the technology. Vias are established between v_D and v or v_U and v when overlaps occur between the lower $(z_v - 1)$ or upper $(z_v + 1)$ layers of the grid. The starting SMT formulation is based on both edge and vertex consumption. We implement detailed routing variables, constraints, and design rule constraints as described in [14]. Specifically, we define routing grids and constraints using commodity flow conservation (CFC), vertex exclusiveness (VE), edge assignment (EA), and edge exclusiveness through metal segment (MS). Design rule constraints are defined employing geometric variables (GV) to prevent violation of min area rule (MAR) and end-of-line (EOL) spacing rule.[2]

D. Objectives in SMT

The open-source SMT solver $Z3$ [23] is equipped with a lexicographic minimization ordering capability. Incorporating the objective within the SMT optimizer serves to not only minimize perturbations from the routed layout but also ensure stability and determinism. Our objective is set as follows: 1) ΔV, 2) ΔH, 3) ΔF, 4) ΔP, and 5) ML{#$VIA12$, #$M2$, #$VIA23$, #$M3$}.[3] Table I shows details of the objective. This setup minimizes the vertical and horizontal movement of cells,

[2]Further details are provided in Section III.A.2 of [14]. Additionally, our actual implementation of the CFC, VE, EA, MS, GV, MAR and EOL rule types is available in Lines 3926-3940, 3942-3971, 3973-4055, 4057-4133, 4173-4206, 4208-4230 and 4232-4260 respectively of EcoBase.cpp in [20].

[3]Further details can be found in Section II.E of [4]. Additionally, our actual implementation of the objectives is available in Lines 4496-4602 of EcoBase.cpp in [20].

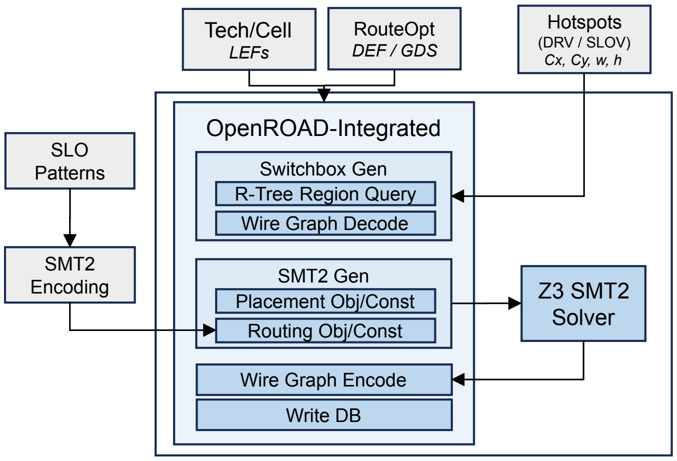
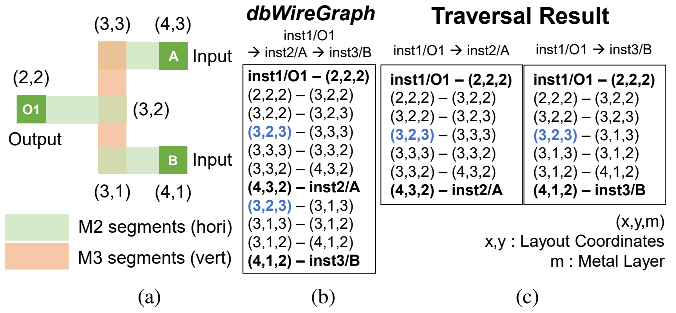


Fig. 4. Our implementation is based on the open-sourced framework within [22]. All steps from R-Tree region query to writing the DB file exist within the single binary, which enables fast in-memory computing.



inst1/O1 – (2,2,2)
(2,2,2) – (3,2,2)
(3,2,2) – (3,2,3)
(3,2,3) – (3,3,3)
(3,3,3) – (3,3,2)
(3,3,2) – (4,3,2)
(4,3,2) – inst2/A
(3,2,3) – (3,1,3)
(3,1,3) – (3,1,2)
(3,1,2) – (4,1,2)
(4,1,2) – inst3/B

inst1/O1 → inst2/A	inst1/O1 → inst3/B
inst1/O1 – (2,2,2)	**inst1/O1 – (2,2,2)**
(2,2,2) – (3,2,2)	(2,2,2) – (3,2,2)
(3,2,2) – (3,2,3)	(3,2,2) – (3,2,3)
(3,2,3) – (3,3,3)	(3,2,3) – (3,1,3)
(3,3,3) – (3,3,2)	(3,1,3) – (3,1,2)
(3,3,2) – (4,3,2)	(3,1,2) – (4,1,2)
(4,3,2) – inst2/A	**(4,1,2) – inst3/B**

Fig. 5. An example of *dbWireGraph* traversal. (a) Routed metal segments in a single net, (b) available one-dimensional list from *dbWireGraph*, and (c) final traversal results.

cell flipping, pin extensions, and the number of routed segments in the VIA12, M2, VIA23, and M3 layers, respectively. Reducing the number of routed segments contributes to a decrease in routed wirelength.

The above discussion has presented the SMT framework baselines for placement (Subsection III-B), routing (Subsection III-C), and objectives (Subsection III-D). The next section describes how we improve on these SMT framework baselines to achieve an automated ECO framework that rectifies both DRC and SLO violations.

IV. Our Proposed Frameworks

Figure 4 illustrates the ECO flow that addresses DRC and SLO violations within each switchbox. The necessary inputs include the tech and cell LEFs, post-routeOpt (i.e., final-routed) layout DEF, hotspot locations, widths, heights, and SLO patterns. Our implemented flow operates within the open-source framework provided by OpenROAD [22], which minimizes file-based communications and facilitates in-memory interactions between an EDA- and physical design-optimized database and P&R tools.

A. Switchbox Generation

Utilizing the physical dimensions of a given switchbox, we employ R-Tree [7] twice, once for intersected instances and the other for intersected nets. All instances and routed metal segments are included in two R-Trees, allowing the extraction of intersected instances and nets based on the 2D physical information (width, height) of the switchbox. After the extraction of related nets, we decode the routed wires using the *dbWireGraph* APIs.

The *dbWireGraph* from OpenDB [18] offers a rapid wire traversal method for routed metal segments, arranged in sorted order from the output pin to various input pins. Using the traversal methods in-built to OpenDB, we extract all the metal segments of a given routed net, and save all *fromPin* to *toPin* pairs found (i.e., during the *dbWireGraph* traveral of the net). Figure 5 shows an example of *dbWireGraph* traversal.

Figure 5(a) and (b) show the wire graph traversal of a three-pin net $\{$*inst1/O1, inst2/A, inst3/B*$\}$ will be initiated from *inst1/O1* and continue to *inst2/A* and *inst3/B* successively. As the routed net can contain multiple Steiner points, the net traversal can encounter multiple branchings, or *branch points*, from the current ongoing path. We store coordinates of all points (i.e., segment endpoints of the routed net) in a hashmap with integer-value triples (x, y, m), where x and y represent layout coordinates, and m indicates a metal layer. In Figure 5(b), $(3, 2, 3)$ will contain two children $(3, 3, 3)$ and $(3, 1, 3)$.

Throughout the traversal with *dbWireGraph*, whenever a metal segment encounters previously visited points from the hashmap, we record the previous point $\rightarrow$ new child point connections to track updated branch points. Upon termination of a metal segment at sink endpoints (*inst2/A* or *inst3/B*), we save the back-tracked paths from the output pin (i.e., source) to the current sink point. In our example, after the traversal procedure, there will be two pairs: $\{$*inst1/O1*$-$*inst2/A*$\}$ and $\{$*inst1/O1*$-$*inst3/B*$\}$. Figure 5(c) shows two lists (i.e., back-tracked paths) after *dbWireGraph* traversal.

Using the obtained $\{$*fromPin*$-$*toPin*$\}$ pairs, we identify the physical locations of all beginning and ending points, as determined by the switchbox boundary. We then trim and eliminate the intermediate metal segments between these beginning and ending points, as depicted in Figure 2; the SMT formulation will handle the routing of these intermediate metal segment parts. The improved runtime achieved through this wire graph decoding and net extraction method is demonstrated below in Subsection V-B.

B. SMT Generation - SLO Constraints

Recall that Subsections III-B and III-C respectively develop the base frameworks for placement and routing constraints. We now explain the addition of SLO patterns via further SMT constraints, elaborating on the procedures for generating SLO constraints in detail.

To compile the SLO-violated patterns obtained from the pattern matching tool, a specific SLO pattern matching rule, two examples of which are depicted in Figure 6(a), must be provided. In this representation, a metal segment's existence is denoted by 1, while its absence is represented by 0. The pattern rule encompasses grids of various widths in units of

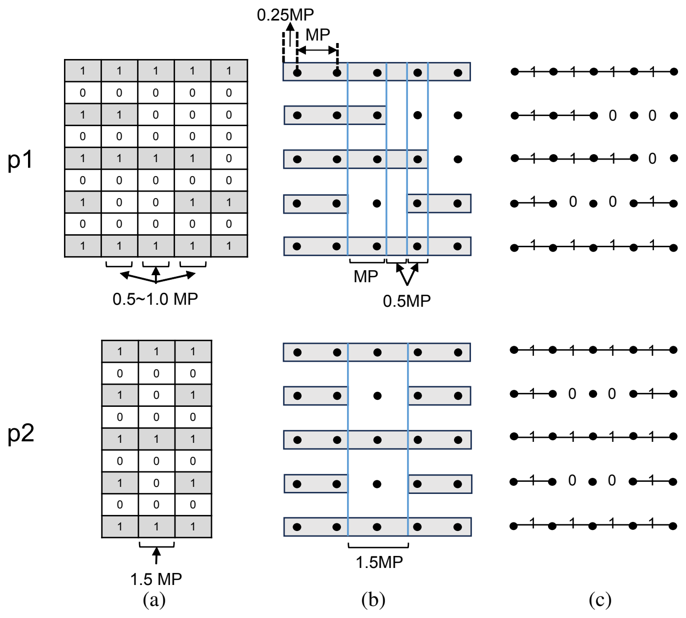


Fig. 6. Two example single-line-open (SLO) patterns, *p1* and *p2*. (a) The SLO pattern matching rules that are defined for the pattern matching tool. (b) An example of the matched layout. (c) Converted SMT constraints within the switchbox. The pattern matching tool determines the blue lines. Note that the 1/0 patterns used in pattern matching rules (a) and the SMT2 constraints (c) are not isomorphic.

metal pitches (MP). For instance, in pattern *p1* of Figure 6(a), the width bounds of the 2nd to 4th columns in the first row are set from $0.5MP$ to $1.0MP$. Figure 6(b) illustrates matched patterns in layout, where the shaded rectangles are metal segments in the layout. The pattern matching tool determines blue boundaries and furnishes the center coordinates of these matched patterns. Then, Figure 6(c) shows the converted SLO pattern constraints. We formulate the SLO constraints for SMT as

$$SLO = \bigwedge_{r \in P_R, c \in P_C} f(m_{r,c}\_m_{r,c+1}) \tag{2}$$

$$f(m) = \begin{cases} m & \text{if } m \text{ is 1 in pattern } P \\ \neg m & \text{otherwise} \end{cases} \tag{3}$$

where $m_{r,c}\_m_{r,c+1}$ denotes horizontal metal segments from (r, c) to $(r, c+1)$ in Figure 6(c), P_R represents the row set of an SLO pattern P, and P_C denotes the column set of an SLO pattern P.

Multiple SLO constraints are necessary to encompass the entire switchbox. In particular, the SLO pattern should not exist anywhere in a given switchbox, so all possible locations of the pattern must be explicitly encoded into constraints. To ensure effective switchbox generation, we confine the dimensions of the switchbox to match the size of the SLO pattern constraints. Consideration is also given to flipped patterns, which include the original pattern as well as mirroring about the x-axis, the y-axis, and both the x- and y-axes. For instance, in the scenario of a switchbox having dimensions of (7,6) metal pitches for *width* and *height*, the total count of SLO constraints is calculated as 2 (*patterns*) * 4 (*flips*) * 4 (*width*; 7-4+1) * 2 (*height*; 6-5+1) = 64, where the pattern's (*width*, *height*) = (4,5) for both *p1* and *p2* in Figure 6.

C. Round of SLO-ECO

Figure 7 provides an overview of a single round within our SLO-ECO framework. Importantly, the earlier work of [4] primarily focuses on addressing DRC violations represented in orange. By contrast, our proposed approach takes into account *both* DRC violations and multiple (in our current discussion and experiments: two) distinct SLO pattern violations. In a single round, DRC checks are performed using a commercial P&R tool, while SLO checks are conducted through a pattern matching tool. Our proposed framework extracts switchboxes and solves SMT instances to address all three of 1) DRC-violated hotspots, 2) *p1* violated hotspots, and 3) *p2* violated hotspots. Given the heightened failure rate of SMT due to increased pin density, our proposed flow, depicted in Figure 4, is executed in a multi-threaded mode. Each thread operates with distinct offset, width, and height dimensions centered around the DRC and SLO violated hotspots. If any thread discovers an optimal routed layout, our framework halts to address the current hotspot and proceeds to resolve subsequent violations. Conversely, if no improvement (i.e., reduction of DRC and SLO violations) is observed, our framework does not move forward with the next round.

V. Experimental Setup and Results

TABLE II
BENCHMARKS FOR SLO-ECO EXPERIMENTS. WE USE VARIOUS TECHNOLOGIES AVAILABLE IN [5].

Design	Technology			#Insts	FP Util	#DRC Vios	#SLOVs		Total
	RT	DRSet	MPO		(%)		p1	p2	#Vios
aes_1	4T	EL	3	13k	73	41	135	47	223
aes_2	4T	EL	2		72	25	132	58	215
aes_3	4T	ET	2		72	78	108	56	242
aes_4	4T	ET	3		73	97	125	40	262
jpeg_1	4T	EL	2	71k	87	156	123	33	312
jpeg_2	4T	EL	3		87	37	143	29	209
jpeg_3	4T	ET	2		90	199	125	31	355
jpeg_4	4T	ET	3		92	50	117	34	201
ldpc_1	4T	EL	3	55k	27	51	209	226	486
ldpc_2	4T	ET	2		28	14	89	126	229
ldpc_3	4T	EL	3		28	104	219	235	558
ibex_1	4T	ET	2	11k	95	147	34	30	211
ibex_2	4T	EL	2		96	93	45	24	162
ibex_3	4T	ET	3		97	60	54	27	141

We have implemented our proposed SLO-ECO frameworks using approximately 11k lines of C++ and SMT-LIB 2.0 format. We describe our experimental setups with 14 benchmarks in Subsection V-A, and we present experimental confirmations of improved runtimes compared with the previous work, and of improved quality of results, in Subsection V-B.

A. Experimental Setup

We run our experiments in CentOS 8 Stream environments with an Intel Xeon Gold 6148 2.40GHz (80T) CPU and 384GB RAM machine. We use Cadence $Innovus$ 21.1 to

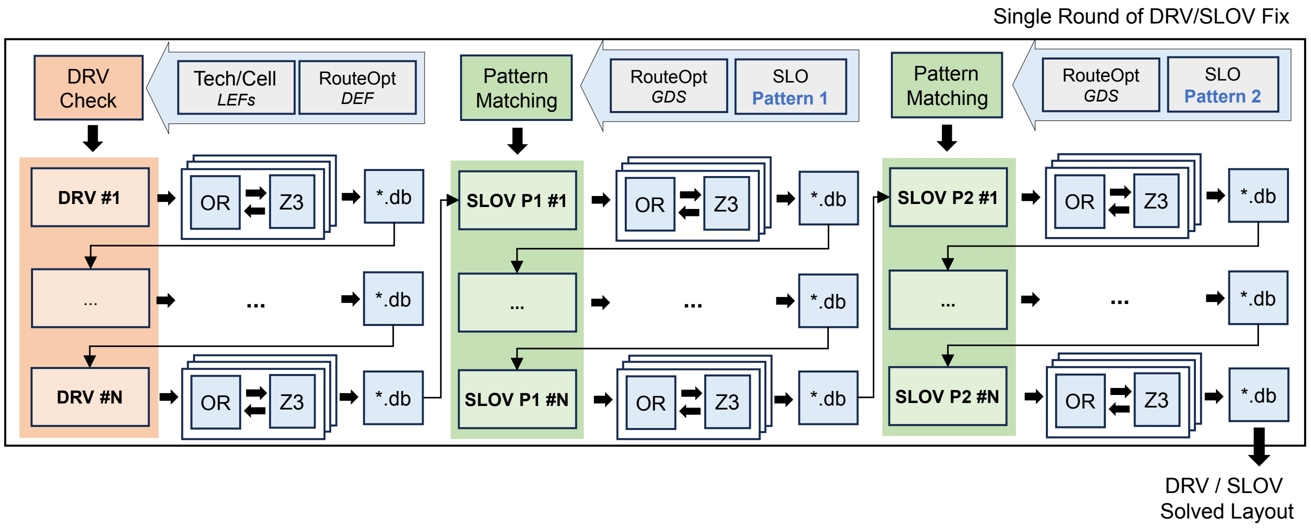


Fig. 7. Single round of our proposed framework. We indicate by OR $\leftrightarrow$ Z3 the box given in Figure 4. Since OpenROAD's OpenDB supports fast database saving and reading functions, our framework can take snapshots in the form of a *.db file. For each single round, our proposed framework extracts switchboxes and solves SMT instances to address all three of 1) DRC-violated hotspot, 2) *p1* violated hotspot, and 3) *p2* violated hotspot.

generate designs and extract the DRC violations. We use the Cadence Physical Verification System [15] - *eClair* 21.1 as the pattern matching tool to extract the SLO violations. We use the SMT solver *Z3* [23] 4.11.2. We limit the number of parallel threads to 16 for each round in our framework and we limit the number of threads to 1 for *Z3*, *Innovus*, and *eClair* tools.

Table II shows 14 publicly available benchmarks used in our experiments, obtained from [16], [17].[4] We use openly-published technology setups that reflect EUV patterning, available in [5]. The three technology parameters RT, DRSet, and MPO respectively denote number of routing tracks, design rule set, and minimum pin opening parameters. FP util indicates the initial floorplan utilization. In the design rule set (DRSet) of technology parameters, "EUV loose" (EL) and "EUV tight" (ET) are distinguished by three key design rules: min-area-rule (MAR), end-of-line (EOL), and via restriction (VR) within a routing grid. We set MAR/EOL/VR = 1/1/1 for EL, and MAR/EOL/VR=1/2/1 for ET, following [5].[5]

TABLE III
SWITCHBOX TRAVERSAL PARAMETERS USED IN OUR FRAMEWORKS.

Category	Parameters used in each round		
	1st	2nd	3rd-5th
swbox_width	6, 8, 10, ..., 18	7, 9, 11, ..., 19	6, 8, 10, ..., 18
swbox_height	6, 10, 14, 18	7, 11, 15, 19	8, 12, 16
dx	-10, -8, -6, ..., 10	-10, -8, -6, ..., 10	-10, -8, -6, ..., 10
dy	-10, -6, -2, ..., 10	-10, -6, -2, ..., 10	-10, -6, -2, ..., 10

Table III shows the switchbox generation parameters used in each round. We set the *max_rounds* = 5 and we provide different parameter sets for the 1st, 2nd, and 3rd-5th rounds to cover the violations in various ways. The physical dimension of the switchbox is determined as *center = (cx+dx*MP, cy+dy*MP)*, *width = swbox_width * MP*, and *height = swbox_height * MP*, where *(cx, cy)* is the center coordinate of a given hotspot and *MP* is the metal pitch of each layer. We set the lower layer as M2, the upper layer as M3, and equal metal pitch for M2 and M3. For each single round, we generate all possible combinations of parameters provided in Table III. For instance, our framework tries 7 * 4 * 11 * 6 = 1848 switchbox for each DRC and SLO violation in the first round. To generate distinct sets of switchboxes in each round, we set the *swbox_height* as $4n + 2$ in the first round, $4n + 3$ in the second round, and $4n$ in the third to fifth rounds where n is an integer.[6] Note that to ensure awareness of both $p1$ and $p2$, the dimensions defined in Table III are strictly larger than the dimensions of both $p1$ and $p2$.

B. Experimental Results

Table IV provides a runtime comparison of switchbox generation with that of the previous work [4]. Our proposed switchbox generation method reduces the runtime by up to more than 1000× for the *ldpc* design, because combining OpenDB with R-Tree region query enables us to keep more operations in working memory. The switchbox generation can finish within 5 seconds for all designs used in Table II.

Table V shows our overall results for handling both DRC and SLO violations together. In the table, *#Re.DRVs*, *#Re.SLOVs* and *#Re.T.Vios* respectively denote the number of remaining DRC violations, the number of remaining SLO violations, and the number of remaining total violations. We calculate the Impr (%) as *(orig-new)/orig*. On average across our testcases, we achieve wirelength reduction of 0.368%,

[4] *aes, jpeg, ldpc* are from [17] and *ibex* is from [16].

[5] Additional description is in [5]. The GitHub repository [21] provides generated standard cells of [5]. Additionally, a recent enhancement of [5] has open-sourced various useful collaterals in [19].

[6] Based on our experience, maintaining the same dimension from the 3rd to 5th rounds has proven effective in reducing DRC violations. The potential impact of different parameter sets remains a subject for future exploration.

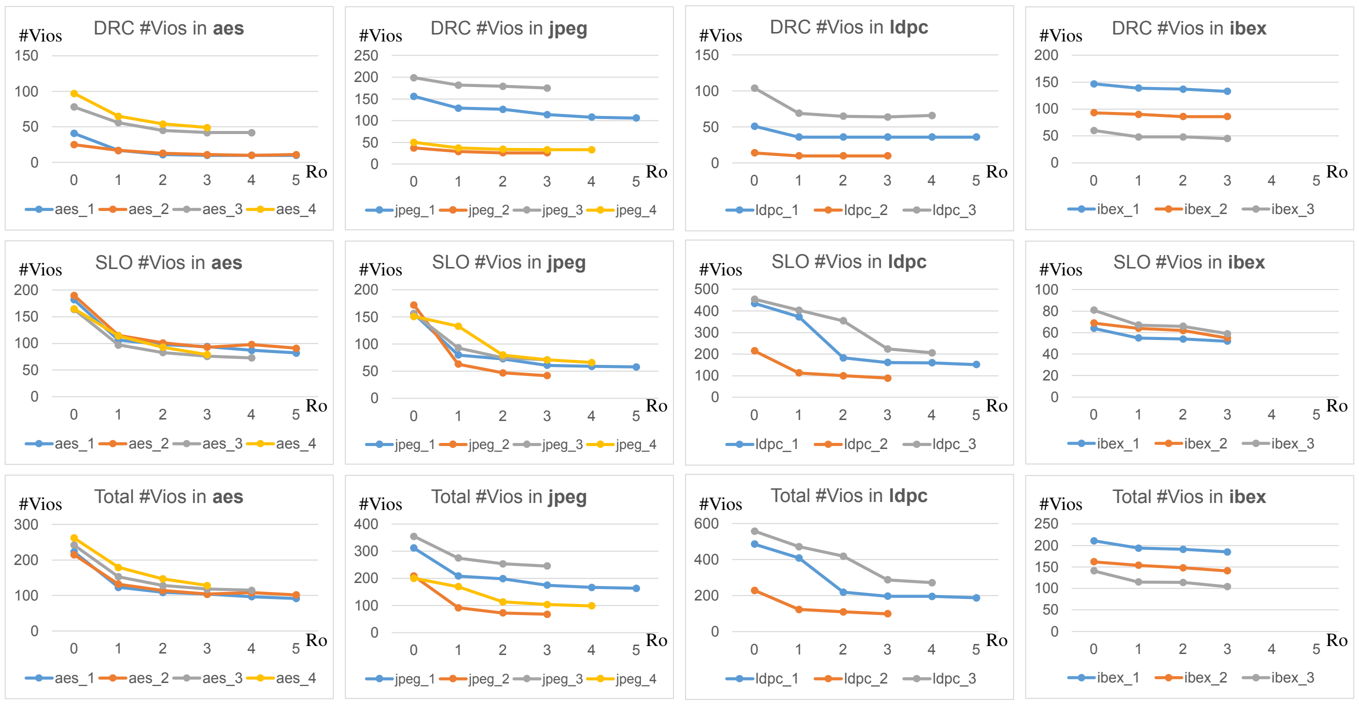


Fig. 8. The trend of remaining DRC, SLO and total violations (in Table V, *#Re.DRVs*, *#Re.SLOVs*, and *#Re.T.Vios*) corresponding to *#ECO_Round* (Ro). We do not proceed with the next round when the *#Re.T.vios* cannot be improved further. We set *max_rounds* = 5.

TABLE IV
RUNTIME COMPARISON OF SWITCHBOX GENERATIONS.

Design	CoRe-ECO [4]	Our Work	
	time (s)	time (s)	Ratio
aes_1	62	1.01	61.38
aes_2	69	1.20	57.50
aes_3	73	1.11	65.76
aes_4	71	1.47	48.29
jpeg_1	1058	1.56	678.20
jpeg_2	966	1.87	516.57
jpeg_3	955	1.63	585.88
jpeg_4	1039	1.62	641.35
ldpc_1	9585	2.93	3271.33
ldpc_2	9519	3.51	2711.96
ldpc_3	9732	2.75	3538.90
ibex_1	91	0.51	178.43
ibex_2	86	1.20	71.66
ibex_3	88	0.79	111.39

DRC violation reduction of 33.68%, SLOV reduction of 54.51% (*p1*) and 43.62% (*p2*), and total violations reduction by 45.14%, within #ECO round = 3.78 and runtime = 15.64 hours.

Figure 9 shows examples of layouts produced by our SLO-ECO framework that fix SLO violations within the *ldpc_1* design. Figure 9(a) shows the first SLO pattern, and (b) shows the re-routed layout using our framework. Figures 9(c) and (d) analogously illustrate the SLO violation fix for a second pattern case. The routed layouts are captured using the OpenROAD GUI infrastructure [22]. We believe that our

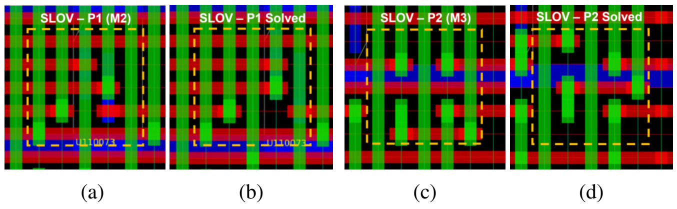


Fig. 9. Improved SLO violations achieved by our SLO-ECO framework, for the *ldpc*_1 testcase. M1 is shown in blue color, M2 in red, and M3 in green. (a) The SLO-violated window of pattern 1 in M2. (b) The solved window of pattern 1. (c) The SLO-violated window of pattern 2 in M3. (d) The solved window of pattern 2.

studies confirm that our SLO-ECO framework is able to solve SLO violations and provide final-routed layout solutions within runtimes that are quite reasonable – particularly in light of automation, scalability to more threads, and the cost/latency of human-expert fixing.

Figure 8 shows the trend of the total number of violations with *#ECO_round* for various designs. We confirm a decreasing trend in the number of total DRC and SLO violations in all 14 designs. Our framework achieves up to 58.74%, 67.46%, 61.31% and 26.24% reduction of total violations on the *aes*, *jpeg*, *ldpc* and *ibex* designs, respectively.

VI. CONCLUSIONS

In conclusion, our work has introduced a novel SMT-based simultaneous P&R methodology, called *SLO-ECO*, which significantly advances the ability to mitigate and avoid DRC violations in automated VLSI physical design, particularly in advanced sub-7nm processes. SLO-ECO integrates simultaneous detailed placement and routing considerations, ef-

TABLE V
OUR RESULTS FOR DRC AND SLO VIOLATIONS OF PATTERN 1 AND PATTERN 2.

Design	Initial					Our Proposed SLO-ECO Method											
	WL(um)	#DRVs	#SLOVs		Total	WL		#Re.DRVs		#Re.SLOVs				#Re.T.Vios		#ECO	Runtime
			p1	p2	#Vios	WL(um)	Incr(%)	#DRVs	Impr(%)	p1	Impr(%)	p2	Impr(%)	#Vios	Impr(%)	Round	(h)
aes_1	45035.544	41	135	47	223	44978.244	-0.127	10	75.60	52	61.48	30	36.17	92	58.74	5	11.49
aes_2	45849.794	25	132	58	215	45668.294	-0.395	11	56.00	62	53.03	29	50.00	102	52.55	5	9.64
aes_3	45968.760	78	108	56	242	45841.970	-0.275	42	46.15	50	53.70	23	58.92	115	52.47	4	20.10
aes_4	45680.790	97	125	40	262	45491.210	-0.415	49	49.48	62	50.40	17	57.50	128	51.14	3	15.34
jpeg_1	134901.914	156	123	33	312	134896.074	-0.004	106	32.05	42	65.85	16	51.51	164	47.43	5	41.35
jpeg_2	133558.226	37	143	29	209	133558.746	+0.000	26	29.72	31	78.32	11	62.06	68	67.46	3	10.58
jpeg_3	132739.504	199	125	31	355	132743.944	+0.003	175	12.06	51	59.20	20	35.48	246	30.70	3	25.62
jpeg_4	135582.560	50	117	34	201	135589.160	+0.004	33	34.00	48	58.97	18	47.05	99	50.74	4	14.24
ldpc_1	761899.755	51	209	226	486	750298.940	-1.522	36	29.41	64	69.37	88	61.06	188	61.31	5	21.73
ldpc_2	737957.049	14	89	126	229	734613.279	-0.453	10	28.57	23	74.15	66	47.61	99	56.76	3	4.10
ldpc_3	751812.136	104	219	235	558	743581.536	-1.094	66	36.53	64	70.77	142	39.57	272	51.25	4	25.56
ibex_1	47042.100	147	34	30	211	47002.000	-0.085	133	9.52	27	20.58	25	16.66	185	12.32	3	8.36
ibex_2	49346.749	93	45	24	162	49279.729	-0.135	86	7.52	37	17.77	18	25.00	141	12.96	3	5.25
ibex_3	45469.447	60	54	27	141	45171.950	-0.654	45	25.00	38	29.62	21	22.22	104	26.24	3	5.65
Average	222346.023	82.28	118.42	71.14	271.85	220622.505	-0.368	59.14	33.68	46.50	54.51	37.42	43.62	143.07	45.14	3.78	15.64

ficient switchbox extraction, SMT utilization, and runtime enhancements to achieve significant reduction in both DRC and SLO violations. Experimental studies across a variety of testcases confirm substantial improvements in post-P&R violation counts, as well as the (first-ever) automated avoidance and mitigation of SLO patterns. SLO-ECO can benefit monotonically from application of additional threads in parallel, and can extend to handle additional SLO and DRC rules. It therefore offers the promise of improved VLSI layout design automation at advanced manufacturing nodes. We leave as directions for future work application of *SLO-ECO* to contexts with multiple distinct metal pitches, addition of more SLO patterns, and exploration of the runtime tradeoffs from adding more SLO patterns.

ACKNOWLEDGMENTS

We thank the authors of [4] for providing their source codes ($\sim$17k lines of Perl code) and permission to open-source our work upon publication. We thank Samsung Electronics for financial support and technical collaboration. Research at UCSD was partially supported by DARPA HR0011-18-2-0032 and the C-DEN center.

REFERENCES

[1] P. D. Bisschop and E. Hendrickx, “Stochastic Printing Failures in EUV Lithography”, *Extreme Ultraviolet (EUV) Lithography X*, SPIE vol. 10957, 2019, pp. 37-56.

[2] P. Cao, G. He, W. Ding, Z. Zhang, K. Wang and J. Yang, “Efficient and Accurate ECO Leakage Optimization Framework With GNN and Bidirectional LSTM”, *IEEE Trans. on VLSI Systems* 31(9) (2023), pp. 1413-1424.

[3] C.-K. Cheng, C. Holtz, A. B. Kahng, B. Lin and U. Mallappa, “DAG-Sizer: A Directed Graph Convolutional Network Approach to Discrete Gate Sizing of VLSI Graphs” *ACM TODAES* 28(4) (2023), pp. 52:1-31.

[4] C.-K. Cheng, A. B. Kahng, I. Kang, M. Kim, D. Lee, B. Lin, D. Park and M. Woo, “CoRe-ECO: Concurrent Refinement of Detailed Place-and-route for an Efficient ECO Automation”, *Proc. ICCD*, 2021, pp. 366-373.

[5] C.-K. Cheng, A. B. Kahng, H. Kim, M. Kim, D. Lee, D. Park and M. Woo, “PROBE2.0: A Systematic Framework for Routability Assessment from Technology to Design in Advanced Nodes”, *IEEE Trans. on CAD* 41(5) (2022), pp. 1495-1508.

[6] V. Dai, J. Yang, N. Rodriguez and L. Capodieci, “DRC Plus: Augmenting Standard DRC with Pattern Matching on 2D Geometries”, *Proc. SPIE* vol. 6521, 2007, pp. 1-12.

[7] A. Guttman, “R-trees: A Dynamic Index Structure for Spatial Searching”, *Proc. ACM SIGMOD*, 1987, pp. 47-57.

[8] K. Han, A. B. Kahng and H. Lee, “Evaluation of BEOL Design Rule Impacts Using an Optimal ILP-based Detailed Router”, *Proc. ACM/IEEE DAC*, 2015, pp. 1–6.

[9] S. Heo, A. B. Kahng, M. Kim, L. Wang and C. Yang, “Detailed Placement for IR Drop Mitigation by Power Staple Insertion in Sub-10nm VLSI”, *Proc. DATE*, 2019, pp. 824-829.

[10] A. B. Kahng, L. Wang and B. Xu, “The Tao of PAO: Anatomy of a Pin Access Oracle for Detailed Routing”, *Proc. ACM/IEEE DAC*, 2020, pp. 1-6.

[11] J. Kim, J. Kim, B. Shin, S. Lee, J.-H. Kang, J.-W. Jeon, P. Pathak, J. Condella, F. E. Gennari, P. Hurat and Y.-C. Lai, “Process Related Yield Risk Mitigation with In-design Pattern Replacement for System ICs Manufactured at Advanced Technology Nodes”, *SPIE Design-Process-Technology Co-optimization for Manufacturability XIV*, vol. 11328, 2020, pp. 116-123.

[12] W. Lee, Y. Kwon and Y. Shin, “Fast ECO Leakage Optimization Using Graph Convolutional Network”, *Proc. GLSVLSI*, 2020, pp. 187-192.

[13] H. -C. Lin and S. -Y. Fang, “MIA-aware Detailed Placement and VT Reassignment for Leakage Power Optimization,” *Proc. ASPDAC*, 2023, pp. 1-6.

[14] D. Park, D. Lee, I. Kang, C. Holtz, S. Gao, B. Lin and C.-K. Cheng, “Grid-Based Framework for Routability Analysis and Diagnosis with Conditional Design Rules”, *IEEE Trans. on CAD* 39(12) (2020), pp. 5097-5110.

[15] Cadence Physical Verification System. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/silicon-signoff/physical-verification-system-for-fast-final-signoff.html

[16] lowRISC: Collaborative open silicon engineering. https://www.lowrisc.org

[17] OpenCores: Open-Source IP Cores. https://opencores.org

[18] OpenDB: Database and Tool Framework for EDA. Dec 16, 2020, https://github.com/The-OpenROAD-Project/OpenDB

[19] PROBE3.0 GitHub. Sep 20, 2023, https://github.com/ABKGroup/PROBE3.0

[20] SLO-ECO GitHub. Apr 3, 2024, https://github.com/ABKGroup/SLO-ECO

[21] SMT-Based Standard-Cell Layout Generator for PROBE2.0. July 22, 2020, https://github.com/ckchengucsd/SMT-based-STDCELL-Layout-Generator-for-PROBE2.0

[22] The OpenROAD Project GitHub. Apr 10, 2023, https://github.com/The-OpenROAD-Project/OpenROAD/

[23] Z3, a theorem prover from Microsoft Research. Apr 10, 2023, https://github.com/Z3Prover/z3