ICMarks: A Robust Watermarking Framework for Integrated Circuit Physical Design IP Protection

Ruisi Zhang, Rachel Selina Rajarathnam, *Senior Member, IEEE*, David Z. Pan, *Fellow, IEEE*, and Farinaz Koushanfar, *Fellow, IEEE*

***Abstract*—Physical design watermarking (WM) on contemporary integrated circuit (IC) layout encodes signatures without considering the dense connections and design constraints, which could lead to performance degradation on the watermarked products. This article presents `ICMarks`, a quality-preserving and robust WM framework for modern IC physical design. `ICMarks` embeds unique watermark signatures during the physical design's placement stage, thereby authenticating the IC layout ownership. `ICMarks`'s novelty lies in 1) strategically identifying a region of cells to watermark with minimal impact on the layout performance and 2) a two-level WM framework for augmented robustness toward potential removal and forging attacks. Extensive evaluations on benchmarks of different design objectives and sizes validate that `ICMarks` incurs no wirelength and timing metrics degradation, while successfully proving ownership. Furthermore, we demonstrate `ICMarks` is robust against two major WM attack categories, namely, watermark removal and forging attacks; even if the adversaries have prior knowledge of the WM schemes, the signatures cannot be removed without significantly undermining the layout quality.**

***Index Terms*—Intellectual property protection (IPP), physical design, watermarking (WM).**

I. Introduction

IN THE modern very-large-scale integrated circuit (VLSI) supply chain, companies across multiple countries collaborate to design, fabricate, assemble, and verify integrated circuits (ICs) [1], [2]. The final product integrates intellectual property (IP) components from various stakeholders, including design [3], [4], [5], [6] and fabrication houses [7], along the global supply chain. With the confluence of diverse inputs, safeguarding IP emerges as an undeniable critical necessity, particularly to preempt IP piracy threats [8], [9], [10]. The physical design of the supply chain bridges the logic design with the VSLI layout for manufacturing. It strategically optimizes the positioning and connectivity of components on the chip canvas and enhances power–performance–area (PPA) metrics. The optimizations not only translate into million-dollar manufacturing cost savings for chip producers [11] but also establish invaluable IPs. However, they are vulnerable to unauthorized use, like forging attacks by malicious third parties in the supply chain [12], and require robust protective measures.

Watermarking (WM) [13], [14], [15] has emerged as a viable methodology to safeguard physical design IP by embedding unique and confidential signatures into the IC layout. To authenticate ownership, the design houses verify the watermark signatures by decoding the watermarks from the IC layout. Existing physical design WM frameworks [12] proposed to insert watermarks from two directions: 1) constraint-based WM [16], [17], [18], [19], [20] and 2) invasive WM [6], [21]. Constraint-based WM encodes signatures by setting cell position/topology as optimization constraints during the physical design. Such frameworks, primarily designed for partition-based placement algorithms [22], [23], do not consider new design constraints in modern IC layouts, such as fence regions, leading to quality degradation. The invasive WM adds redundant cells [6] or wires [21] as watermarks. However, the signatures could be forged [24] if the adversary has knowledge of the WM algorithms.

Inserting watermarks into modern IC layouts is nontrivial and exhibits several challenges. First, large-scale designs have dense standard cell connections, and slight perturbations of the cell locations/topologies can degrade the overall performance. Second, modern ICs often have additional design constraints like fence regions and macros. Developing WM algorithms without considering these additional design constraints will negatively affect overall performance on certain layouts. Finally, the watermarks should be robust against potential removal and forging attacks from the supply chain.

This article devises `ICMarks`, a novel and robust WM framework to safeguard IC layout IP. It consists of two consecutive steps, namely, global WM (GW) and detailed WM (DW), and targets the watermark insertion at the global and detailed placement stages, respectively. GW leverages a novel scoring scheme to identify a region to watermark that does not violate the design constraints and ensures minimal performance deterioration. The identified watermarked region, along with the associated cells inside it, encodes the GW signature by co-optimizing the placement such that only the associated cells intersect the watermarked region. Next, DW selects cells that do not overlap with nearby cells when moving along the x/y-axis within GW's watermarked region. The

Received 8 March 2024; revised 4 August 2024 and 29 January 2025; accepted 12 March 2025. Date of publication 18 March 2025; date of current version 22 September 2025. This work was supported by the NSF TILOS AI Institute under Award 2112665. This article was recommended by Associate Editor I. H.-R. Jiang. *(Corresponding author: Farinaz Koushanfar.)*

Ruisi Zhang and Farinaz Koushanfar are with the Department of Electrical and Computer Engineering, University of California at San Diego, San Diego, CA 92122 USA (e-mail: ruz032@ucsd.edu; fkoushanfar@ucsd.edu).

Rachel Selina Rajarathnam and David Z. Pan are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA (e-mail: rachelselina.r@utexas.edu; dpan@ece.utexas.edu).

Digital Object Identifier 10.1109/TCAD.2025.3552503

cells are watermarked by shifting in the x/y directions to encode DW signature in the detailed placement. As such, the strategically inserted watermarked region and cells ensure the signature insertion satisfies the modern IC design constraints and incurs minimal quality degradation. The design company proves ownership by reverse engineering [25], [26] the logic netlist and all standard cell locations from an IC layout in GDSII format, and employ ICMarks to decode signature for ownership verification.

ICMarks is robust against threats from the supply chain that aim to remove or forge the watermarks. By encoding signatures as placement co-optimization objectives, it can withstand a spectrum of watermark removal attacks [12]. Extensive targeted watermark removal attacks in different layout regions validate that ICMarks still maintains over 90% watermark extraction rates (WERs) when the adversaries compromise a maximum of ∼2% wirelength/timing quality. In addition, the two-level augmented GW and DW signatures make the watermark strength stronger compared with the best prior practice [19], [20], and forging the signatures becomes exceedingly harder. The WM strength is measured by the probability that the watermark constraints are satisfied by coincidence.

In brief, our contributions are as follows.

1) We present ICMarks, the first WM scheme with both region and position constraints for the modern VLSI layouts, which does not degrade the layout quality while being robust against watermark removal and forging attacks.
2) Our framework features: a) a novel search algorithm to identify the watermarked region that adheres to design constraints and incurs minimal performance degradation in GW and b) an innovative encoding mechanism for quality-preserving signature insertion and augmented robustness in DW.
3) We conduct experiments on benchmarks of different design objectives and sizes: a) ICMarks introduces no degradations on the wirelength-driven ISPD'2015 [27] and ISPD'2019 [28] benchmarks, whereas the best prior method [5] degrades the average routed wirelength by up to 1.4% and b) ICMarks shows no degradations on the timing-driven ICCAD'2015 [29] benchmarks, whereas the best prior method [19], [20] degrades the total and worst negative slack (WNS) by 1.14% and 1.51%, respectively.[1]
4) We perform comprehensive robustness analysis: a) ICMarks withstands watermark removal attacks targeting different layout regions and maintains over 90% WERs and b) ICMarks resists watermark forging attacks, making it hard to counterfeit the signatures.

II. Background and Related Work

In this section, we introduce the background for VLSI placement and IC design WM.

[1]The IC physical design is fragile that 0.5% performance degradation can compromise the additional efforts from the design company for quality metrics optimization [30].

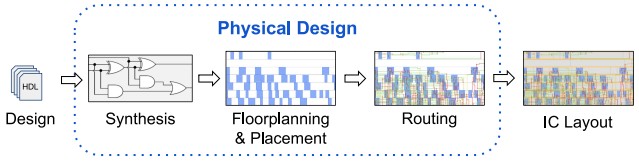


Fig. 1. VLSI physical design process. It generates the IC layout from a design RTL through synthesis, placement, and routing stages.

A. VLSI Placement

The IC design flow starts with design specifications that are converted to register transfer level (RTL) using a hardware description language (HDL). As depicted in Fig. 1, the physical design process synthesizes the design HDL to generate a logic-level netlist comprising macros and standard cells from the technology library. A floorplanning step determines regions and locations for the design's macros and the IO (input–output) ports. Typically, the standard cells are placed during the placement stage, followed by routing all the design connectivity. The routed design is verified to ensure manufacturability and the physical layout is generated in GDSII format to send to the fabrication unit for manufacturing.

With fixed macro locations, placement is an essential step in the physical design flow to determine the locations of all the standard cells in the design [30], [31], [32], [33]. It significantly impacts the quality and efficiency of the subsequent stages, including routing and manufacturing. The VLSI placement typically consists of three stages: 1) global placement; 2) legalization; and 3) detailed placement.

1) Global Placement: In the global placement stage, the placer aims to achieve roughly legal cell locations by distributing the cells across the chip area to minimize overlaps. In addition, the placer also targets other objectives, such as minimizing wirelength and ensuring timing constraints are met.

Wirelength-Driven Placement: A wirelength-driven placer aims to minimize the overall wirelength W while ensuring minimal cell overlap, as specified in the following equation [34]:

$$\min_{x,y} \; W(x, y) \; s.t. \; D(x, y) < \mathcal{D} \tag{1}$$

where (x, y) is the location of cells in the placement, and $W(x, y)$ denotes the total wirelength connecting cells in the design. $D(x, y)$ is a density measure of the overlaps among the cells, and it should be below the threshold $\mathcal{D}$.

Timing-Driven Placement: A timing-driven placer prioritizes optimizing timing critical paths in the design while ensuring the overlaps among the cells are minimum [35]. Given a timing endpoint t (a primary output port or an input pin of memory element) with an arrival time of $t_{\text{AT}}(t)$ and a required arrival time $t_{\text{RAT}}(p)$, the *slack* of t is calculated as follows:

$$\begin{aligned} \text{slack}(t) &= t_{\text{RAT}}(t) - t_{\text{AT}}(t) \\ \text{TNS} &= \sum_{t \in P_{\text{end}}} \min(0, \text{slack}(t)) \\ \text{WNS} &= \min_{t \in P_{\text{end}}} \text{slack}(t). \end{aligned} \tag{2}$$

With the timing endpoints P_{end} in the design, the total negative slack (TNS) and the worst negative slack WNS are computed as shown in (2). A timing-driven placer optimizes either the WNS or the *TNS* to meet the timing constraint of the design.

2) Legalization: The legalization stage moves cells to eliminate the cell overlaps, ensure the row alignments, and guarantee the design constraints are met [36], [37], [38]. The multirow height cells are prioritized for legalization before the single-row height cells.

3) Detailed Placement: The detailed placement stage refines the legalized placement to improve design metrics, such as wirelength or timing constraints [39], [40]. The refinement can be achieved by operations like 1) swapping the positions of two cells to improve the objectives without causing legality or timing violations and 2) window-based approaches to find optimal cell locations within the specified region.

B. Watermarking in IC Design

WM [13], [14], [15] encodes unique and confidential signatures into the IC layouts to assist product owners in proof of ownership. Design houses leverage this technique to enhance IP rights protection in the chip supply chain and detect unauthorized usages or replications. These watermarks are typically inserted in the logic design or physical design level.

1) Logic Design Watermarking: The WM at the logic design level focuses on safeguarding the RTL or netlist ownership. Designers embed unique signatures into the RTL or netlist that do not impact the core functionality and are invisible upon adversaries' inspection. The signature insertion can be categorized as follows: 1) modifying the finite state machine (FSM) to add additional inputs or unused components as watermarks [3], [4]; 2) adding additional power components as side-channel signatures [41], [42]; and 3) introducing unique triggers during netlist design that induces the malfunctions when the triggers are detected [43].

More recent works introduce logic locking to add additional logic gates and circuits that alter the original design's functionality when the incorrect key is applied [44], [45]; obfuscation to adding dummy logic gates or additional circuit connections to make the logic design appear differently from the original one [46], [47]. These techniques are orthogonal to the logic design WM, which can be combined to establish more robust IP protection (IPP) frameworks.

Logic Design WM Faces Several Limitations:

1) Adding additional watermarks into the logic design needs to consider physical design criteria like timing to maintain chip quality. However, WM after these constraints have been optimized at the physical design stage reduces such discrepancies.
2) In logic design, if the logic circuits are not performing correctly after watermark insertion, designers need to revisit the RTL design in the previous stage. Physical WM, however, does not modify the circuits' functionalities and only changes how cells are placed/connected on the canvas.

TABLE I
COMPARISON OF IC WM FRAMEWORKS. ○ FRAMEWORK DOES NOT MEET CRITERIA; ◐ FRAMEWORK PARTIALLY MEETS CRITERIA; ● FRAMEWORK COMPLETELY MEETS CRITERIA

Stage	WM Method	Fidelity	Efficiency	Robustness
Logic Design	Logic Design WM [3], [41], [43]	◐	○	◐
Physical Design	Invasive WM [21], [6]	◐	●	○
	Constraint-based WM [19], [20], [16], [17], [18]	◐	●	●
	ICMarks	●	●	●

2) Physical Design Watermarking: The WM at the physical design level encodes signatures onto the IC layout from two directions.

1) Constraint-based WM [5], [16], [17], [18], [19], [20] that uses cell topology/position as the additional constraints during physical design. Row Parity [19], [20] constraints 1-bit watermark cells to be on the odd row and the 0-bit watermark cells to be on the even row. Cell Scattering [5] enforces the 1-bit watermark cell to move one unit along the y-axis and 0-bit cell one unit along the x-axis.
2) Invasive WM [6], [21] that adds redundant cells or wires as the watermark signatures.

The WM frameworks' quality is evaluated by the following metrics.

Criteria 1—Fidelity: The watermarked layouts shall incur minimal quality degradation compared with nonwatermarked ones on metrics like wirelength and timing. Besides, the watermarked layouts should adhere to the layouts' design constraints, such as fence regions and row alignments.

Criteria 2—Efficiency: The WM framework shall be efficient with minimal overheads for watermark insertion.

Criteria 3—Robustness: The watermarked layout shall be robust against various attacks targeting to remove or forge the signature.

The constraint-based WM frameworks [16], [17], [18], [19], [20] encodes signatures during the placement stage of physical design. The watermarks are embedded as part of the layout. By encoding the constraints into the layout, constraint-based WM are robust against removal or forging attacks. The overhead of watermark insertion remains negligible compared to the time required to optimize the IC layout. However, the WM frameworks do not consider the new design constraints, such as macros and fence regions that could lead to performance degradation.

By adding redundant cells/wires, the invasive watermark [6], [21] is efficient for watermark insertion, and the quality degradation depends on how the corresponding algorithm designs the insertion mechanism. Nevertheless, the encoded watermarks can be forged if the adversaries have knowledge of the WM algorithm.

III. PROBLEM FORMULATION

In this section, we introduce ICMarks's goal and its threat model.

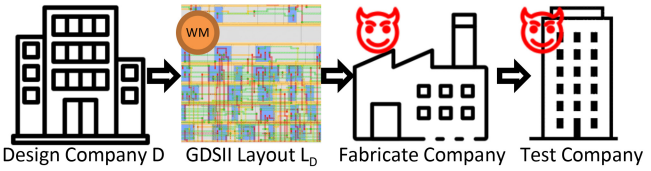


Fig. 2. IC layout watermark scenario in the supply chain. The design company watermarks the IC layout before manufacturing and testing.

WM Goal: In the physical design stage, the IC design company invests huge efforts in identifying and fine-tuning optimization objectives for better cell placement and net routing. The optimization and customization are infused into different levels of the final layout, including (but not limited to) blocks and cells, to boost IC performance and reduce manufacturing expenses. As such, the final IC layouts constitute invaluable IP, emphasizing the need to safeguard it.

ICMarks establishes ownership proof for the VLSI physical design by inserting watermarks into the placement stage, thereby authenticating the IP for the final layout. Fig. 2 depicts a typical scenario within the supply chain. The IC design company D uses highly customized and optimized placement and routing algorithms to enhance physical design quality. The layout L_D from the physical design stage is watermarked and sent to the fabrication company for manufacturing. The manufactured ICs are verified for their functionalities in the test company. The design company D can employ reverse engineering approaches [25], [26] to extract the logic netlist and all standard cell locations from an IC layout in GDSII format, and employ ICMarks to decode the signature for ownership verification.

Threat Model: The adversary $\mathcal{A}$ in the fabricate or test company aims to steal the layout and stop D from claiming its ownership by executing different attacks.

Adversary's Locations: We consider the adversary $\mathcal{A}$ to be in the fabrication or test company and has access to the layouts produced by the design company.

Adversary's Objective: Adversary $\mathcal{A}$'s objective is to 1) prevent design company D from ownership proof by removing or forging the encoded signature and 2) the attacked layout performance, such as wirelength or timing metrics, shall not be significantly compromised.

Adversary's Capacities: We consider an adversary $\mathcal{A}$ with the following capacities: 1) the adversary $\mathcal{A}$ has access to the layout L_D. However, he cannot reverse-engineer the optimizations and customizations to reproduce L_D because they are infused into every level of the design and 2) the adversary $\mathcal{A}$ has access to open-source physical design tools to remove or forge the watermark. He also knows the general algorithms to watermark the layout. However, the random seeds, signatures, and insertion parameters are beyond his reach.

IV. ICMarks Design

This section presents the proposed layout WM ICMarks, a two-level scheme that inserts watermarks during multiple stages of VLSI placement. In Section IV-A, we introduce the GW method. Next, we propose an independent embedding of the watermark signatures as DW in Section IV-B. While GW and DW can be used independently for IP protection, ICMarks combines both techniques to take advantage of their best properties.

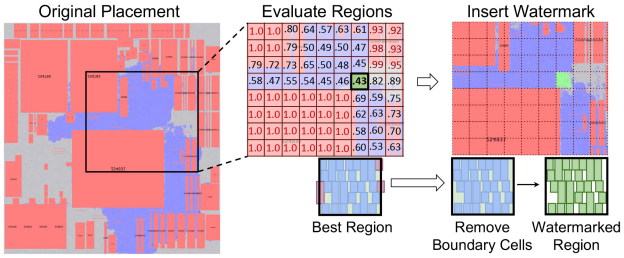


Fig. 3. GW pipeline with stride = sliding window size. The blue cells are the standard cells, the red cells are the macros, and the green cells are wm cells in GW.

A. Global Watermarking

GW embeds watermarks during the global placement stage. The watermarks, including a predefined watermarked region and associated cells in the region, are embedded as a co-optimization term during global placement. The co-optimization objective is to ensure only those associated cells are placed in the watermarked region.

1) GW Watermark Selection: Given the original, optimized placement denoting the position of all cells as P_{ori}, we employ a sliding window-based algorithm that traverses the placement to search for a region to watermark as shown in Fig. 3. An ideal watermarked region should meet three criteria: 1) as the signatures are encoded by moving cells along the x/y-axis in DW, the number of cells N_c within the region should be sufficient to accommodate the required number of signature bits N_w; 2) the total cell area S_{cell} within the watermark region are S shall be small, thereby providing ample space for watermarked cells to maneuver. The small watermark region utilization ensures the watermarks are encoded in the low-density regions; and 3) the cells area S_{overlap} overlap on the watermarked region boundary shall be minimized, such that their displacement from the watermarked region has minimal impact on the layout performance. These requirements are incorporated into an evaluation function f, as illustrated in (3), to evaluate each region of the original placement P_{ori}

$$f = \alpha \frac{N_w}{N_c} + \beta \frac{S_{\text{cell}}}{S} + \gamma \frac{S_{\text{overlap}}}{S}. \tag{3}$$

The scoring function is balanced by adjusting the weights α, β, γ, where $\alpha, \beta, \gamma \in [0, 1]$. The scores of the traversed regions are then normalized into a range of [0, 1]. For regions that are nested within macros or other fence regions and any region where $N_c < N_w$, the evaluation function f yields the highest evaluation scores as 1.0, rendering the regions not suitable for inserting GW watermarks. The region with the minimum score is chosen as our target watermarked region R_w, and the set of associated cells within the region is denoted as C_{w1}.

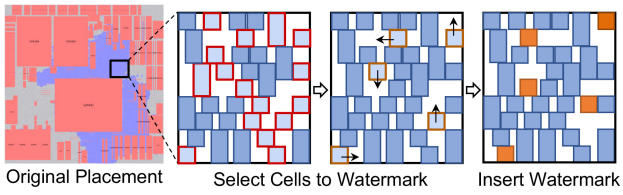


Fig. 4. DW pipeline with candidate cells from which watermarked cells are selected.

2) GW Watermark Insertion: The watermarked region R_w and its associated cells C_{w1} is formulated as the additional watermarked region constraint in the placement objective. It enforces the associated cells C_{w1} to be in the watermarked region R_w and other cells to be out of R_w. For a design with K fence regions, the placement formulation includes watermarked region R_w as the additional optimization constraints, as specified in

$$\min_{v} \sum_{e \in E} W(e; v) + \lambda D(v)$$
$$\text{s.t.} \quad v_k = (x_k, y_k) \in R_k, \quad k = 0, \ldots, R_K, R_w. \tag{4}$$

W is the wirelength term and D is the cell density term with density multiplier λ. v denotes the (x, y) location of cell and $e \in E$ is the design net. The watermarked region R_w is added as an additional region constraint with associated cells C_{w1} to obtain a watermarked placement P_w.

3) GW Watermark Extraction: The design owner asserts ownership by employing the watermarked region R_w to extract cells C'_{w1} within the region and comparing these with the associated cells C_{w1} and nonassociated cells C_{w1o}. The extraction rate of the watermark, denoted as WER_{GW}, is calculated in

$$\%\text{WER}_{\text{GW}} = 100 \times \frac{|(C'_{w1} \cap C_{w1}) - (C'_{w1} \cap C_{w1o})|}{|C_{w1}|}. \tag{5}$$

B. Detailed Watermarking

In DW, the watermarked cells are embedded by moving them slightly along the x- or y-axis after legalization. A follow-up detailed placement stage compensates for the performance degradation from the watermark insertion.

1) DW Watermark Selection: The DW moves cells along the x- or y-axis to insert watermarks into the design. Randomly selecting watermarked cells and moving them without considering the dense interconnection could lead to significant performance degradation. To avoid this, DW only selects cells that will not overlap with their neighbors after cell movements as the watermark as shown in Fig. 4.

Algorithm 1 outlines how DW identifies cells that will not overlap with neighbors during WM movements and subsequently inserts the signature. It starts by comparing the positions of cells within each row and identifying the cell indices that can move $|d_x|$ along the x-axis. Then, the algorithm compares cells across adjacent rows to determine if they can be moved up or down for $|d_y|$. The cells C_x with room along the x-axis and C_y with room along the y-axis are identified as candidate cells for WM. D_x and D_y are associated with these candidate cells to retain a record of their movement directions and distances.

Algorithm 1: DW Watermark Selection and Insertion

Require: Legalized placement (X_{lg}, Y_{lg}), Signature B_N, Unit distance $|d_x|$ and $|d_y|$
Ensure: Watermarked cells C_{w2}, Placement (X_{wm}, Y_{wm})

```
Initialize empty sets C_x, C_y, D_x, D_y
for r = 0 to total_num_rows do
    for c in cells_in_row(r) do
        if is_movable_x(c) then
            Append c to C_x
            s = get_direction_x(c) // left: -1; right: 1
            Add (s × |d_x|) to D_x
        if is_movable_y(c) then
            Add c to C_y
            s = get_direction_y(c) // down: -1; up: 1
            Add (s × |d_y|) to D_y
Random shuffle C_x and C_y
X_itr, Y_itr = X_lg, Y_lg
for i = 0 to |B_N| do
    if B_N[i] == 1 then
        c_i = random_choose(C_x) // no replacement
        X_itr[c_i] = X_lg[c_i] + D_x[c_i]
    else
        c_i = random_choose(C_y) // no replacement
        Y_itr[c_i] = Y_lg[c_i] + D_y[c_i]
    Add c_i to C_w2
X_wm, Y_wm = detailed_placement(X_itr, Y_itr)
return C_w2, X_wm, Y_wm
```

2) DW Watermark Insertion: At the end of legalization with placement $P_{\text{lg}} = (X_{\text{lg}}, Y_{\text{lg}})$, cells are moved along the x- or y-axis to insert unique watermark signatures hashed as N bit sequences B_N. If the ith bit in the signature B_N is 1, the cell c_i randomly selected from the candidate watermarked cell set C_x is moved along the x-axis for $D_x[c_i]$. If the ith signature bit is 0, the cell c_i is randomly chosen from the candidate watermarked cell set C_y and moved along the y-axis for $D_y[c_i]$. The resulting placement is denoted as intermediate placement $P_{\text{itr}} = (X_{\text{itr}}, Y_{\text{itr}})$, and the indices of the cells moved along the x- or y-axis form the watermarked cells C_{w2}. These movements are performed before detailed placement, allowing possible performance degradation to be compensated during the subsequent detailed placement phase and yield a watermarked solution $P_{wm} = (X_{wm}, Y_{wm})$. The selected watermarked cells C_{w2}, and their corresponding position distance $\text{Dist} = P_{\text{itr}}(C_{w2}) - P_{wm}(C_{w2})$ between intermediate placement $P_{itr}(C_{w2})$ and the watermarked placement $P_{wm}(C_{w2})$ constitute the watermark. The random wm cell selection before detailed placement ensures the signatures are encoded randomly.

3) DW Watermark Extraction: The design owner detects watermarks in the placement P' by comparing the watermarked cell C_{w2} position with the intermediate placement P_{itr} and calculates the new distance Dist' as $P_{\text{itr}}(C_{w2}) - P'(C_{w2})$. If C'_{w2} cells in the extracted Dist' matches the Dist both along the x- and y-axis, the WER is calculated as

$$\%\text{WER}_{DW} = 100 \times \frac{|C'_{w2}|}{|C_{w2}|}. \tag{6}$$

C. ICMarks Watermarking

As a combination of both the GW and the DW, ICMarks applies GW during its global placement stage and DW on top

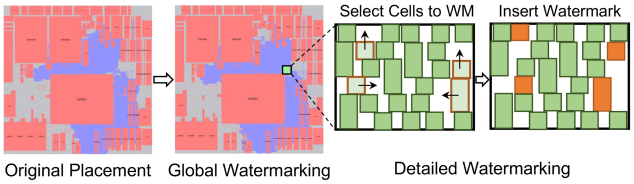


Fig. 5. ICMarks framework. ICMarks first applies GW during its global placement and then applies DW on top of the watermarked region before the detailed placement. The green cells are the wm cells in the GW stage and orange cells are the wm cells in the DW stage.

of the watermarked region before its detailed placement stage. By combining these two WM schemes, the inserted watermark is further augmented.

1) ICMarks Watermark Insertion: As shown in Fig. 5, ICMarks employs a hybrid approach that combines GW and DW to produce stronger watermarks. Initially, ICMarks searches for a watermarked region using the strategy akin to GW, yielding a watermarked region R_w and associated cells C_{w1}. ICMarks runs global placement and legalization with such watermark constraints. Then, it perturbs cells along the x- and y-axis within the watermarked region R_w, following the approach of DW and obtain an intermediate placement P_{itr}. Afterward, the detailed placement proceeds to compensate for WM degradation and produce the watermarked placement P_{wm}. The perturbed distance Dist and cells C_{w2} are the watermarks in the second step.

2) ICMarks Watermark Extraction: To claim ownership of the layout, given the cell positions P', the watermark holder can use the following two steps: 1) the extraction of the watermarks C'_{w1} from the watermarked region R_w, with the global extraction rate denoted as in (5) and 2) afterward, the inserted signatures are extracted by comparing $P' - P_{itr}$ with Dist at the watermarked cell indices C_{w2}. The number of cells successfully matched is denoted as C'_{w2}, from which the detailed extraction rate is calculated as in (6). The final WER is calculated in

$$\%\text{WER}_{\text{ICMarks}} = 100 \times \frac{\text{WER}_{GW} + \text{WER}_{DW}}{2}. \quad (7)$$

V. Experiments

In this section, we first introduce the experiment setups in Section V-A. Then, we evaluate the performance of wirelength-driven and timing-driven design WM in Section V-B. Next, we show the robustness of ICMarks under attacks in Section VI.

A. Experimental Setup

Datasets: We evaluate the WM impact on wirelength-driven benchmark suites: ISPD'2015 [27] and ISPD'2019 [28]; and the timing-driven ICCAD'2015 [29] benchmark suite. All designs in the ICCAD'2015 benchmark suite are large and contain more than 700k cells. The statistics and node technology of these designs are summarized in Table II, with designs having fence regions marked in blue. The macros, fence regions, and IO pin locations in the benchmarks are fixed.

TABLE II
Benchmark Statistics and Node Technology. The Designs With Fence Regions Are in Blue. The Sliding Window's Size and Stride Are Multiples of the Adjacent Row-Height

Suite	Design	Cells	Nets	Size	Stride	Node
ISPD 2015 (WL)	perf_a	108K	115K	10	5	65nm
	perf_b	113K	113K	10	5	65nm
	dist_a	127K	134K	20	5	65nm
	mult_b	146K	152K	10	5	65nm
	mult_c	146K	152K	20	5	65nm
	pci_a	30K	34K	10	5	65nm
	pci_b	29K	33K	20	5	65nm
	superblue11	926K	936K	100	40	28nm
	superblue16	680K	697K	50	10	28nm
	perf_1	113K	113K	50	10	65nm
	fft_1	35K	33K	10	5	65nm
	fft_2	35K	33K	10	5	65nm
	fft_a	34K	32K	10	5	65nm
	fft_b	34K	32K	20	5	65nm
	mult_1	160K	159K	50	10	65nm
	mult_2	160K	159K	50	10	65nm
	mult_a	154K	154K	50	10	65nm
	superblue12	1293K	1293K	50	10	28nm
	superblue14	634K	620K	50	10	28nm
	superblue19	522K	512K	100	10	28nm
ISPD 2019 (WL)	ispd19test1	9K	3K	10	5	32nm
	ispd19test2	73K	72K	10	5	32nm
	ispd19test3	8K	9K	10	1	32nm
	ispd19test4	151K	146K	10	5	65nm
	ispd19test5	29K	29K	10	10	65nm
	ispd19test6	181K	180K	50	10	32nm
	ispd19test7	362K	359K	50	25	32nm
	ispd19test8	543K	538K	50	10	32nm
	ispd19test9	903K	895K	200	20	32nm
	ispd19test10	903K	895K	10	5	32nm
ICCAD 2015 (Timing)	superblue1	1209k	1215k	100	50	45nm
	superblue3	1213k	1224k	100	30	45nm
	superblue4	795k	802k	100	50	45nm
	superblue5	1086k	1100k	100	50	45nm
	superblue7	1931k	1933k	100	30	45nm
	superblue10	1876k	1898k	100	30	45nm
	superblue16	981k	999k	100	30	45nm
	superblue18	768k	771k	100	50	45nm

Experiment Setup: ICMarks is implemented with Python 3.9 and benchmarked on a Linux Ubuntu system equipped with NVIDIA TITAN XP GPUs, each with 12-GB RAM, and 48 Intel Xeon CPUs, accompanied by 128-GB RAM. The WM methodology of ICMarks can be integrated into any VLSI physical design framework, and we chose to build upon the open-source placement framework, DREAMPlace [48] to demonstrate its viability. The DREAMPlace [48] accelerates the state-of-the-art placement algorithm ePlace [49]/RePlAce [50] on a GPU and maintains the same level of performance. For routing, we employ the open-source state-of-the-art framework CUGR [51]. For timing-driven benchmark suite, we use DREAMPlace 4.0 [35] where the timing metrics are measured by OpenTimer [52].

Hyperparameters: For the hyperparameters, we perform a grid search for the GW phase. In the search of (3), the $\alpha = \{0.1, 0.5, 0\}$, $\beta = \{0.1, 0.5, 0\}$, and $\gamma = 0.1$. The search terminates when the watermarked layout quality is not compromised. For all the considered benchmark suites, we report the GW hyperparameters in Table II. In the DW phase, $d_x = 1$, and d_y is set to one adjacent row-height.

Baseline: We choose the following state-of-the-art WM frameworks as our baselines.

1) Topology constraint-based *Row Parity* [19], [20] that inserts unique bit sequences as watermarks by shifting cells to different rows in the placement stage. Cells with a 1-bit are moved to an odd row, while cells with a 0-bit are moved to an even row.
2) Position constraint-based *Cell Scattering* [5] that employs pseudorandom coordinate transformation (PRCT) algorithms to scatter the watermarked cells on the chip canvas as watermarks. Cells with 1-bit are moved along the y-axis, and cells with 0-bit are moved along the x-axis if they do not overlap with their neighbors.
3) Invasive *Buffer Insertion* [6] that adds additional buffers as watermarks without affecting timing critical paths during the placement stage. Two buffers are inserted to represent 0-bit, and one buffer is inserted to represent 1-bit. For fair comparisons, we reimplemented the baselines [5], [6], [19], [20] and integrated them into the DREAMPlace codebase for benchmarking.

We skip the baselines that: 1) have different WM targets. References [17] and [18] are designed for smaller full-custom IC designs, and [53] and [54] are designed for FPGAs, whereas our WM targets are modern digital VLSI designs and 2) have similar WM approaches as our baselines, and we use the baselines as a proof of concept. Reference [16] inserts flip-flops instead of buffers into the layouts as watermarks. The signature length is set to 50-bit for all frameworks.

Evaluation Metrics: For the wirelength-driven benchmark suites, whose quality is measured by the layout wirelength, we use three metrics to evaluate the watermark performance.

1) *Placement Wirelength Rate (PWLR)*: The rate of estimated half-perimeter wirelength (HPWL) of watermarked layout compared to the original one.
2) *Routing Wirelength Rate (RWLR)*: The rate of routed wirelength of watermarked layout compared to the original one,
3) *WER*: The percentage of signatures extracted from the watermarked layout.

For the timing-driven benchmark suite, whose quality is measured by the timing slack, we use three metrics to evaluate the performance under static timing analysis [52].

1) *TNS Rate (TNSR):* The rate of TNS of the watermarked layout compared to the original one.
2) *WNS Rate (WNSR):* The rate of WNS of the watermarked layout compared to the original one.
3) *WER:* The percentage of signatures extracted from the layout. The rate average is calculated by the geometric mean of designs' metrics.

To preserve the optimized outcomes from the physical design algorithms, we set a threshold of layout quality degradation to not exceed 0.5%. Because surpassing this threshold would counteract the benefits derived from the performance enhancement efforts in physical design [30].

B. Experimental Results

1) Wirelength-Driven Watermarking Fidelity: The performance of different WM schemes is tabulated in Table III for the ISPD'2015 benchmarks [27], and in Table IV for the ISPD'2019 benchmarks [28]. For all the layouts in Tables III and IV, their inserted watermarks can be successfully extracted. Therefore, we evaluate how much layout performance is compromised to accommodate such watermark insertion among these WM frameworks.

Comparison With Prior Constraint-Based WM [5], [19], [20]: Among the two benchmark suites, `ICMarks` results in no PWLR and RWLR degradation for accommodating the 50-bit signature. In contrast, the topology constraint-based Row Parity [19], [20] changes the row index of the cells to encode watermarks but does not consider potential design constraints (e.g., fence regions and macros) in VLSI design. As a result, it results in 2.31% and 1.94% routed wirelength (RWLR) degradation on ISPD'2015 [27] and ISPD'2019 [28], respectively. The position constraint-based Cell Scattering [5] perturbs cell locations in the optimized layout for signature insertion. However, randomly selecting and moving watermarked cells after the optimization is done results in quality degradation, as reflected by the 0.12% and 1.4% RWLR degradation over the original design on ISPD'2015 [27] and ISPD'2019 [28].

Comparison With Invasive WM [6]: The invasive WM Buffer Insertion [6] overlooks the additional design constraints, like fence regions and macros, in the modern VLSI design. The additional 50 buffers might be added close to such design constraints and result in significant cell displacement, as the design has to accommodate the buffers while satisfying the design constraints. As shown in Tables III and IV, Buffer Insertion [6] introduced 9.01% and 8.71% RWLR degradation on ISPD'2015 [27] and ISPD'2019 [28] benchmarks. `ICMarks`, however, has no wirelength degradation on both benchmarks. Besides, for the highly utilized designs, where most of the layout space is filled with standard cell and fixed marcos/fence regions/IO pins, to accommodate the additional buffers, cells have to be displaced significantly from their nonwatermarked position for buffer insertion. As such, it introduced significant PWLR and RWLR degradations as the gray numbers in Table III.

DW Insertion Before/After Detailed Placement: After the placement stage is finished, the baseline Cell Scattering [5] moves cells in the layout along the x/y-axis if there is space. The key difference is Cell Scattering [5] performs the movement after detailed placement, whereas the `ICMarks`: DW is performed before detailed placement. As seen in Tables III and IV, `ICMarks`: DW improves the RWLR quality from Cell Scattering [5] by 0.06% and 1.39%, respectively.

Comparison With the GW and DW Submodules: As a combination of GW and DW, `ICMarks` takes advantage of their strength and further reduces the performance degradation from accommodating watermarks. In contrast to GW, `ICMarks` slightly improves the placement quality. This improvement stems from modifying cell positions within a designated region, either across different rows or along the x-axis. Such modifications inherently disturb solutions formulated during global placement. Given that these disturbances occur in less compact regions, they provide different inputs to the detailed placement process, thereby offering potential optimization trajectories to minimize overall wirelength. The optimization

TABLE III
PERFORMANCE ON THE ISPD'2015 BENCHMARKS [27]. ALL THE DESIGN WATERMARKS ARE SUCCESSFULLY EXTRACTED, I.E., WER = 100%. THE PWLR AND RWLR ARE THE PLACEMENT AND ROUTED WIRELENGTH RATES OVER THE ORIGINAL DESIGNS. FR IS FENCE REGION. THE RESULTS IN GRAY FAIL BUFFER INSERTION WM WITH SIGNIFICANT DEGRADATION ON THE HIGH-UTILIZED DESIGNS (DENOTED WITH ⋆)

Design	Utl.	Row Parity [20], [19]		Cell Scattering [5]		Buffer Insertion [6]		ICMarks: GW		ICMarks: DW		ICMarks: GW+DW	
		PWLR ↓	RWLR ↓	PWLR ↓	RWLR ↓	PWLR ↓	RWLR ↓	PWLR ↓	RWLR ↓	PWLR ↓	RWLR ↓	PWLR ↓	RWLR ↓
perf_a ⋆	71.69%	1.0045	1.0155	0.9978	0.9967	1.5289	2.3270	0.9976	0.9873	0.9994	1.0003	**0.9972**	**0.9873**
perf_b	49.71%	1.0058	1.0267	1.0020	1.0001	1.0176	1.0745	0.9930	0.9901	0.9990	1.0002	**0.9890**	**0.9901**
dist_a	61.62%	1.0015	0.9999	0.9984	0.9965	1.0995	1.1072	1.0004	1.0052	0.9998	1.0005	**1.0004**	**1.0052**
mult_b ⋆	77.31%	1.0047	1.0199	0.9991	0.9923	1.8966	2.9374	1.0010	1.0028	1.0002	1.0044	**0.9994**	**1.0028**
mult_c ⋆	77.31%	1.0031	1.0252	0.9980	1.0023	1.6003	2.7459	1.0017	0.9979	0.9990	1.0020	**0.9963**	**0.9979**
pci_a	50.84%	1.0080	1.0340	0.9931	0.9846	1.6269	1.6849	1.0361	1.0048	1.0011	1.0021	**0.9997**	**0.9950**
pci_b	54.77%	1.0069	1.1173	0.9912	0.9977	1.2239	1.8207	0.9951	1.0026	0.9997	1.0008	**0.9951**	**1.0026**
superblue11 ⋆	73.64%	1.0052	1.0344	1.0278	1.0358	1.6930	3.7086	0.9992	0.9992	0.9995	1.0005	**0.9986**	**0.9992**
superblue16 ⋆	74.49%	1.0030	1.0210	0.9988	0.9989	1.1023	1.1377	1.0018	1.0204	**1.0000**	**1.0000**	1.0017	1.0204
perf_1 ⋆	90.57%	1.0035	1.0199	0.9985	0.9983	1.0150	1.0642	1.0031	0.9973	1.0042	1.0019	**0.9964**	**0.9973**
fft_1 ⋆	83.54%	1.0017	1.0260	1.0167	1.0156	1.0680	1.1457	0.9671	0.9673	1.0026	0.9975	**0.9671**	**0.9673**
fft_2	49.97%	1.0050	1.0260	1.0148	1.0114	1.4603	1.4476	0.9767	0.9770	1.0027	0.9996	**0.9767**	**0.9770**
fft_a ⋆	74.02%	1.0121	1.0200	1.0024	0.9933	1.8203	2.1923	0.9939	0.9895	1.0075	1.0094	**0.9939**	**0.9895**
fft_b ⋆	78.01%	1.0018	1.0075	0.9961	1.0030	1.9859	2.5615	0.9909	0.9890	0.9991	1.0036	**0.9909**	**0.9890**
mult_1 ⋆	80.24%	1.0050	1.0238	1.0077	1.0065	1.0464	1.0631	0.9753	0.9744	1.0004	1.0008	**0.9753**	**0.9744**
mult_2 ⋆	79.03%	1.0033	1.0199	0.9984	0.9967	1.0736	1.1147	0.9867	0.9900	0.9990	0.9992	**0.9852**	**0.9900**
mult_a ⋆	86.10%	1.0037	1.0105	0.9995	0.9968	1.3862	1.6738	0.9973	0.9916	1.0005	0.9958	**0.9973**	**0.9916**
superblue12	57.62%	1.0031	1.0067	0.9979	0.9956	1.0683	1.1044	1.0036	0.9732	0.9994	0.9942	**0.9854**	**0.9732**
superblue14 ⋆	77.63%	1.0020	1.0057	0.9991	0.9981	1.0212	1.0286	**0.9851**	0.9867	1.0001	1.0000	0.9887	**0.9867**
superblue19 ⋆	81.51%	1.0025	1.0077	1.0001	1.0005	1.0295	1.0672	0.9881	0.9809	1.0003	1.0001	**0.9814**	**0.9809**
Average: FR	-	1.0047	1.0322	1.0006	1.0005	1.0724	1.1061	1.0028	1.0011	0.9997	1.0012	**0.9975**	**1.0000**
Average: All	-	1.0043	1.0231	1.0018	1.0012	1.0536	1.0901	0.9946	0.9913	1.0007	1.0006	**0.9908**	**0.9908**

TABLE IV
PERFORMANCE ON ISPD'2019 BENCHMARKS [28]. ALL THE DESIGN WATERMARKS ARE SUCCESSFULLY EXTRACTED, I.E., WER = 100%. THE PWLR AND RWLR ARE THE PLACEMENT AND ROUTED WIRELENGTH RATES OVER THE ORIGINAL DESIGNS

Design	Utl.	Row Parity [20], [19]		Cell Scattering [5]		Buffer Insertion [6]		ICMarks: GW		ICMarks: DW		ICMarks: GW+DW	
		PWLR ↓	RWLR ↓	PWLR ↓	RWLR ↓	PWLR ↓	RWLR ↓	PWLR ↓	RWLR ↓	PWLR ↓	RWLR ↓	PWLR ↓	RWLR ↓
ispd19test1 ⋆	83.14%	1.0060	1.0129	0.9978	0.9998	1.0394	1.0619	0.9995	1.0015	1.0023	1.0043	**0.9955**	**1.0015**
ispd19test2 ⋆	84.11%	1.0184	1.0201	1.0096	1.0016	1.0127	1.0408	**0.9981**	0.9999	1.0021	1.0003	0.9988	**0.9999**
ispd19test3 ⋆	88.77%	1.0130	1.0475	1.0180	1.0193	1.0582	1.0801	1.0059	1.0045	**0.9961**	**0.9976**	1.0059	1.0045
ispd19test4	65.90%	1.0017	1.0050	0.9999	0.9998	1.1452	1.2494	0.9957	0.9907	0.9998	1.0028	**0.9957**	**0.9907**
ispd19test5	43.06%	1.0102	1.0556	1.0998	1.0975	0.9859	1.0121	1.0013	0.9998	**0.9996**	**0.9968**	1.0013	0.9998
ispd19test6 ⋆	74.60%	1.0032	1.0106	0.9994	0.9993	1.0044	1.1108	1.0023	1.0026	**1.0001**	**0.9997**	1.0023	1.0026
ispd19test7 ⋆	95.56%	1.0028	1.0160	1.0136	1.0109	1.0003	1.0717	1.0050	1.0054	**1.0000**	**0.9997**	1.0050	1.0054
ispd19test8 ⋆	79.90%	1.0001	1.0082	0.9966	0.9958	1.0118	1.0806	0.9961	0.9929	0.9999	0.9996	**0.9961**	**0.9929**
ispd19test9 ⋆	84.02%	1.0072	1.0108	1.0108	1.0095	1.0164	1.0814	1.0023	1.0023	**1.0009**	**0.9999**	1.0023	1.0023
ispd19test10 ⋆	88.48%	1.0025	1.0090	1.0043	1.0108	1.0378	1.0982	0.9972	0.9967	0.9999	1.0002	**0.9972**	**0.9966**
Average	-	1.0065	1.0194	1.0060	1.0140	1.0304	1.0871	1.0003	0.9996	1.0001	1.0001	**1.0000**	**0.9996**

TABLE V
PERFORMANCE ON ICCAD'2015 BENCHMARKS [29]. ALL THE DESIGN WATERMARKS ARE SUCCESSFULLY EXTRACTED, I.E., WER = 100%. THE TNSR AND WNSR ARE THE TOTAL AND WNSRS OVER THE ORIGINAL DESIGNS

Design	Utl.	Row Parity [20], [19]		Cell Scattering [5]		Buffer Insertion [6]		ICMarks: GW		ICMarks: DW		ICMarks: GW+DW	
		TNSR ↓	WNSR ↓	TNSR ↓	WNSR ↓	TNSR ↓	WNSR ↓	TNSR ↓	WNSR ↓	TNSR ↓	WNSR ↓	TNSR ↓	WNSR ↓
superblue1	78.07%	1.0053	1.0094	1.0119	1.0026	1.0083	1.0020	0.9001	**0.8514**	1.0256	1.0249	**0.8283**	1.0021
superblue3	76.78%	1.0053	1.0176	1.0066	0.9891	1.1035	1.0988	**0.9302**	**0.9029**	1.0035	0.9884	0.9363	0.9047
superblue4	80.12%	1.1299	0.9914	0.9698	1.0191	1.1954	1.1995	**0.9037**	**0.9370**	0.9554	0.9930	0.9221	1.0094
superblue5	73.30%	0.9801	**0.9887**	1.0126	0.9977	1.0932	1.1174	0.9159	1.0001	0.9922	0.9959	**0.9159**	1.0508
superblue7	76.48%	0.9801	0.9932	0.9979	1.0212	0.9933	1.0199	**0.9584**	0.9758	0.9934	1.0199	0.9636	**0.9727**
superblue10	75.80%	1.0099	**1.0041**	1.0218	1.0141	1.0194	1.2918	1.0069	1.0072	**1.0058**	1.0068	1.0069	1.0072
superblue16	79.23%	0.9814	1.1271	0.9067	1.3274	1.0109	1.0877	1.0166	1.0473	**0.9060**	1.3485	1.0069	**1.0072**
superblue18	67.44%	1.0068	0.9965	0.9981	1.0064	1.0438	1.0613	0.9337	**0.9403**	1.0014	1.0002	**0.9254**	0.9449
Average	-	1.0114	1.0151	0.9900	1.0424	1.0566	1.1063	0.9448	**0.9559**	0.9848	1.0418	**0.9366**	0.9867

to GW depends on the quality of the perturbations introduced, where the refinement on ISPD'2019 [28] is marginal, and the improvement on ISPD'2015 [27] is 0.05% over GW.

Compared with DW, ICMarks watermarks upon the placement solution from GW, whereas DW watermarks the original solution. The additional co-optimization of the GW watermarked region makes the placer further refine the cell positions in the watermarked region and, thus, improves the watermarked layout quality. By encoding DW signature on the GW layout, ICMarks introduced no quality degradations. In contrast, DW watermarks upon the original placement solution, which could introduce more performance degradation when inserting DW signatures. It is reflected by the 0.06% and 0.01% RWLR improvement over the original design on ISPD'2015 [27] and ISPD'2019 [28] benchmarks, respectively.

Layout Utilization and Design Constraints Impact on WM Performance: As shown in Tables III–V, invasive WM yields worse performance on high layout utilization designs ($\geq$ 70%), where the layout utilization is larger than 70%. For invasive WM, the performance degradation becomes worse on designs with additional design constraints, e.g., fence regions

and macros. Because the buffers can be encoded close to such constraints and result in significant cell displacements. In contrast, constraint-based WM approaches co-optimize the watermark and placement objectives to preserve the layout quality.

2) Timing-Driven Watermarking Fidelity: As depicted in Table V, ICMarks continuously preserves the layout quality in timing-driven WM benchmark ICCAD'2015 [29]. By strategically searching for the watermarked region and cells with minimal impact on the performance, ICMarks results in no WNS and TNS degradation. On the opposite, the constraint-based Row Parity [19], [20] and Cell Scattering [5] overlook design constraints in modern VLSI design and did not design the WM algorithms with the timing optimization objectives. It results in 1.51% and 4.24% WNSR degradation over the original designs, respectively. Furthermore, while Buffer Insertion [6] encodes additional buffers on the nontiming critical path, the inserted buffers still result in 5.66% TNSR and 10.63% WNSR timing metrics degradation.

3) Watermarking Capacity: The WM capacity is measured by the maximum length of watermark bits that can be inserted into the layout without significantly degrading the layout quality. For Row Parity [19], [20], Cell Scattering [5], and ICMarks: DW, the signature length corresponds to the moved/inserted cell number. For Buffer Insertion [6], the length is the number of buffer-inserted nets. For ICMarks: GW, the signature length corresponds to the N_w in (3). For ICMarks, the signature length corresponds to the N_w cells in GW and moves N_w cell over the watermarked region in the DW.

We use ISPD'2019 [28] as the benchmarking target and display the results in Fig. 6. As seen, the maximum bits that can be inserted by Row Parity and Cell Scattering are both less than 30 bits. Both methods move the cells across rows after the detailed placement as watermarks, which leads to worse RWLR than PWLR, as the router has to cross rows to connect the cells. Buffer Insertion [6] inserts signatures without considering the design constraints, leading to significant cell displacement after signature insertion. Therefore, it exhibits low WM capacity on ISPD'2019 [28]. The ICMarks: DW's capacity is larger and reaches ∼100 bits. Because it moves the cells before detailed placement and the subsequent detailed placement compensates for the watermark insertion. ICMarks: GW and ICMarks further outperform the scheme. They both demonstrate a capacity of more than 200 bits by exploring the less compact region for watermark encoding. Since ICMarks builds the WM scheme on top of ICMarks: GW, the two PWLR and RWLR lines are close in Fig. 6.

Besides, we analyze the maximum number of watermark regions that can be encoded onto the layout without compromising the quality. We choose the best-scored k regions in the GW stage and encode the k regions during GW. As shown in Fig. 7, encoding three watermark regions results in over 20% wirelength degradation. When encoding two or more region constraints into global placement, more cells are on the selected watermark region boundary. Expelling the boundary cells to ensure the design adheres to WM constraints results in significant cell displacement and degrades the watermarked layout quality.

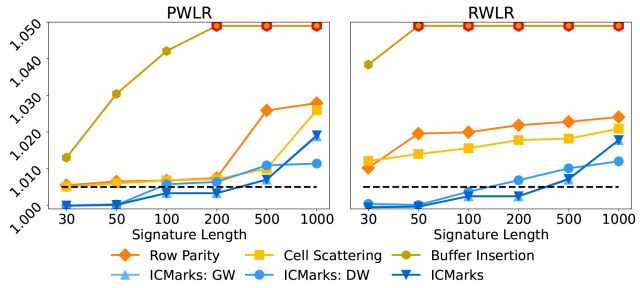


Fig. 6. WM capacity for different frameworks on ISPD'2019 benchmarks [28]. We consider the threshold for acceptable degradation layout quality as 0.5%. The red indicates PWLR and RWLR are higher than the 5% limit.

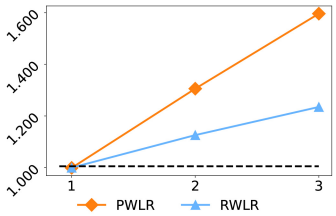


Fig. 7. WM capacity for watermark regions on ISPD'2019 benchmarks [28].

TABLE VI
ICMARKS'S EFFICIENCY ON DIFFERENT BENCHMARKS

Design	non-WM Time (s)	WM Time (s)	Slow Down (%)
ISPD'2015 [27]	141.37	157.96	11.73%
ISPD'2019 [28]	946.23	996.18	9.21%

4) Watermarking Efficiency: The WM efficiency is benchmarked by the fraction of the additional time takes for ICMarks's watermark insertion compared to the overall physical design stage, primarily placement and routing. We include the average time takes to encode 50-bit signatures onto the ISPD'2015 benchmark [27] and ISPD'2019 benchmark [28] in Table VI. The non-WM time and WM time is the average (geometric-mean) time it takes for placement and routing on the nonwatermarked and watermarked layout. The Slow Down is the percentage overhead ICMarks introduced to the overall placement and routing phase. As seen, the time taken for watermark insertion is ∼10% compared with the overall placement and routing stage, making ICMarks efficient for WM. Besides, no additional computation resources or external tools are required for the signature insertion.

5) Watermarking Stealthiness: We display the layout watermarked by ICMarks with various sizes and different design constraints in Fig. 8. As seen, the watermarks are embedded as part of the layout, and invisible upon inspection while maintaining 100% WERs.

6) Watermark Strength: Watermark strength measures the probability a nonwatermarked layout carries the signature by coincidence. For each design with a total of $|C|$ cells, we watermark $|C_w|$ cells and produce the watermarked layout. In the $|C_w|$ cells, x of the cells do not meet the WM constraints. $|C_n|$ from the nonwatermarked layout matches the watermark

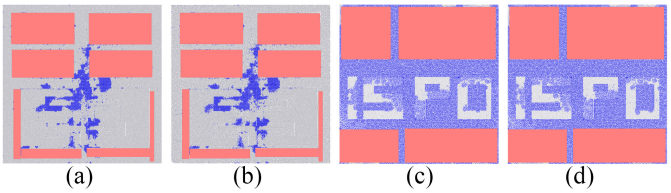


Fig. 8. Watermarked design examples. The blue cells are the standard cells, and the red cells are the macros. (a) test5(WM). (b) test5(no-WM). (c) perf_a(WM). (d) perf_a(no-WM).

TABLE VII
WM STRENGTH FOR DIFFERENT SIGNATURE LENGTHS BY THE PROPOSED ICMARKS WM FRAMEWORK

	Row Parity	ICMarks: GW	ICMarks: DW	ICMarks: GW + DW
P_c	8.8×10^{-16}	9.09×10^{-53}	8.08×10^{-62}	7.35×10^{-114}

requirement by co-incidence. Following Row Parity [19], the watermark strength is calculated by (8) as follows, where $p = (|C_n|/|C_w|)$ is the constraints met by coincidence

$$P_c = \sum_{i=0}^{x} \cdot \mathbf{C}(|C_w|, i) \ \cdot (p)^i \cdot (1-p)^{x-i}. \tag{8}$$

For each design, we calculate the P_c by running the placement with and without ICMarks to obtain $|C_w|$ and $|C_n|$. In most of the designs, $|C_n| = 0$, meaning the nonwatermarked layouts do not match the WM constraints. Therefore, we report the maximum P_c on ISPD'2019 benchmark [28] in Table VII for ICMarks and its submodules. The watermarks are successfully encoded during the global/detailed placement stage and have different positions from their nonwatermarked ones. As such, the signatures' randomness will not be undermined by the global/detailed placement optimizations.

7) Ownership Proof in Real-World Settings: To prove ownership, the design company obtains the suspicious layout and employs reverse-engineering approaches [25], [26] to acquire the logic netlist and all standard cell locations. Such methodology [25], [26] recovers large layouts netlists (over 7000k cells) from the GDSII layout with over 98% accuracy and efficiency. Then, the design company uses the netlists to recover the standard cells' and macros' locations. The watermark extraction algorithms in Section IV are subsequently used for ownership proof. While the reverse-engineering misalignment might degrade the WERs slightly, ICMarks still provides sufficient ownership proof for the design companies benefiting from two aspects: 1) high WM strength of 7.35×10^{-114} for 50-bit signature and 2) high WM capacity that can accommodate more than 200-bit signatures without significant quality degradation, as shown in Fig. 6. As a result, even if only 88% signatures are extracted (10% lost from attacks in Section VI and 2% lost from reverse engineering [25], [26]), ICMarks still provides strong watermarks for 50-bit signatures. In Null Hypothesis [55], a WM strength (p-value) of smaller than 0.05 indicates the statistically significant presence of a watermark. Given the much lower WM strength than 0.05 after reverse engineering, the design company can thereby confidently claim ownership of the design layout.

8) Impact on Routability: We analyze the impact of ICMarks on the design routability using the ISPD'2019 [28] benchmarks employing OpenROAD [56]'s global and detailed router. The final wirelength after detailed routing (Rout. WL), average routing congestion (Rout. Congestion), an initial count of DRC violations (#DRV), the number of detailed routing optimization iterations to fix DRV (#Opt.), and the total valid via access points required for all standard cells and macros in the design (#ValidViaAp) are plotted in Fig. 9 for the watermarked layouts over the nonwatermarked ones.

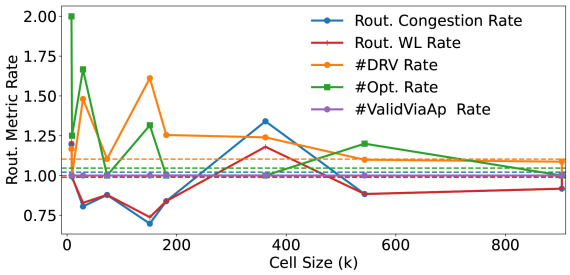


Fig. 9. Impact of WM on routability. The dotted lines represent the geo-mean of large designs ispd19test7–ispd19test10 with $> 300k$ cells.

Across all the considered benchmarks, the watermarked ones have 6% lower Rout. Congestion and Rout. WL compared to the nonwatermarked designs. For large layouts with $> 300k$ cells, ICMarks introduces 2% more routing congestion than nonwatermarked ones, showing it has minimal impact on layout routability. Nevertheless, ICMarks does incur an increase in #DRV and #Opt., by 19% and 20%, respectively, across all the designs for watermarked ones, and 10% more #DRV requiring ~5% more #Opt. for the large designs ispdtest7–ispdtest10 consisting of $> 300k$ cells. #ValidViaAp, representing the layout's pin accessibility, remains similar for watermarked and nonwatermarked designs. Thus, ICMarks does not degrade the pin access of the design.

VI. ATTACK EVALUATION

In this section, we evaluate ICMarks's robustness under watermark removal and forging attacks.

1) Watermark Removal Attack: A successful watermark removal attack shall meet two criteria: 1) the encoded signatures are removed as reflected by the WER below 90% [19], [20] and 2) the layout quality is not compromised, where the placement HPWL rate (PWLR), RWLR, TNSR, and WNSR do not exceed 1.005 [30]. The thresholds are reflected by the black dotted lines in Figs. 10 and 11.

For an IC layout watermarked by ICMarks, the watermarks are inserted by constraining the cell positions and cells' region. To remove the signature, the adversary perturbs the cell positions/regions for watermark removal attacks. We compare four types of attacks aiming to erase the signatures at different levels of WM and prevent the owner from claiming ownership: 1) swap location attacks (SLAs) [19], which target to attack Row Parity framework by random swapping cell locations; 2) constraint perturbation attacks (CPAs), which target to attack Cell Scattering and ICMarks: DW by moving cells along the x/y-axis if there is space; 3) Optimization attacks (OAs), which target to attack all WM frameworks by running another round of detailed placement; and 4) adaptive region attacks (ARAs), which target to attack ICMarks: GW and ICMarks framework by searching for less compacted regions and perturb cells around the region. We show ICMarks

is robust against all removal attacks, and the results are summarized in Figs. 10 and 11 for wirelength-driven and timing-driven placements, respectively.

We skip Buffer Insertion [6] and the corresponding watermark removal attacks because 1) Buffer Insertion's [6] watermarked layout quality degradation is significantly higher than the considered threshold of 0.5% as shown in Tables III–V and 2) other WM frameworks insert signatures without modifying the netlist, whereas Buffer Insertion [6] encodes adding buffers into the netlist as watermarks. While the buffer removal attacks, which remove multiple cascaded buffers, can potentially erase Buffer Insertion's signature, our primary goal is evaluating the robustness of ICMarks under attacks. Therefore, we did not include the attacks at the netlist level in this section.

SLAs: In SLA [19], cells randomly exchange their locations with another set of cells. Then, a follow-up legalization and detailed placement compensate for the performance degradation and ensure the cells follow the design rules. In Fig. 10(a) and (b), we randomly choose 0.1% and 0.5% cells from the layout and pair them to exchange their locations. As seen, even small location swaps result in huge performance degradation on modern IC layouts. In terms of the watermark extraction, Row Parity [19], [20], Cell Scattering [5], and ICMarks: DW get WERs below 90% and failed to verify their ownership. Those WM frameworks spread the watermarks across the layout, where minor changes will subsequently modify most cell locations within the compact region. By embedding cells in the less compact region, ICMarks: GW and ICMarks are less sensitive to such changes. As a result, ICMarks: GW and ICMarks still maintain high WERs.

CPAs: In CPA, cells shift their location along the x-axis for $\delta_x = 1$ or y-axis for δ_y set to one adjacent row-height if such movements do not result in overlapping with their neighbors. In Fig. 10(c)–(e), we move the positions of 0.1%, 1%, and 10% of the cells have space to move in the layout. From here, we find that under CPA 0.1% attack, layout performance hit the boundary of the wirelength degradation (PWLR and RWLR) threshold. For the watermark extraction, ICMarks, ICMarks: GW, and Row Parity [19], [20] achieve over 90% WER. However, the WER of Cell Scattering [5] and ICMarks: DW are subsequently lower than 90% because the constraint perturbation targets to remove the potential watermarked cell positions.

For the CPA 1% and 10% cell performance, however, the PWLR and RWLR degradations are greater than 1.005, meaning the attack failed to maintain the layout quality improvement from the design company's physical design optimizations. But ICMarks: GW and ICMarks still maintain higher than 90% WER and successfully help the design company to claim ownership of the layout.

OAs: This attack employs an additional optimization stage to remove the watermarks. The optimization is implemented through another round of detailed placement. It aims to change cell locations slightly for signature removal while maintaining the layout quality. As shown in Fig. 10(f), the layout wirelength degradation in Cell Scattering [5], ICMarks: GW, ICMarks: DW, and ICMarks are all below 1.005, indicating the attack preserves the layout quality. The WER of Cell Scattering [5] and ICMarks: DW are below 90%, meaning OA successfully removed their signatures. In contrast, ICMarks: GW and ICMarks have over 90% WER, demonstrating their resiliency.

ARAs: This attack targets to remove watermarks in ICMarks:GW. The adversary has prior knowledge of how ICMarks:GW performs the WM and has access to the hyperparameters used to search the watermarked region. The adversary operates on top of the watermarked layout. He tries to remove the inserted watermarks by moving cells within the searched top-1 or top-5 regions if there is room. In Fig. 10(g) and (h), the PWLR and RWLR degradation for ICMarks: GW, ICMarks: DW, and ICMarks are around the threshold of 0.5% degradation, and the attacks do not significantly degrade the layout quality. The WER of ICMarks: GW and ICMarks remain over 90%. As the watermark insertion is performed on a nonwatermarked layout, and the attack regions searched by ARA are on a watermarked layout, the watermark signatures are not the same. Therefore, ARA fails to remove ICMarks: GW and ICMarks's signature, even if the same region WM algorithm is employed. In contrast, the watermarks of Row Parity [19], [20], Cell Scattering [5], and ICMarks: DW are spread across the layout, where minor changes in the compact area will subsequently modify the inserted watermarks. Therefore, these frameworks have compromised WER.

Attacks on Timing-Driven Placement: In Fig. 11, the attack performance on timing-driven placement follows a similar trend as the wirelength-driven placement results in Fig. 10. ICMarks and ICMarks: GW remain resilient under all the removal attacks with a WER over 90%. Row Parity [19], [20] is also resilient to the attacks and has WER of over 90%. However, the watermarked layouts degrade the timing metrics (TNSR and WNSR) by $\geq$1% in Table V for timing-driven placement. The signatures in the baseline Cell Scattering [5] and ICMarks: DW are removed with WER $<$90%.

2) Watermark Forging Attack: Instead of removing the watermarked signature, the adversary in a watermark forging attack counterfeits another set of watermarks on the watermarked layout and falsely claims his ownership. Row Parity [19], [20] and Buffer Insertion [6] techniques are not resilient to forging attacks. If the adversary has prior knowledge of the WM algorithm, they can easily counterfeit a different set of forged signatures from the row index ID of cells or add additional buffers into the layout for false ownership proof. For Cell Scattering [5], signatures are harder to counterfeit because forging the signature requires both random seeds and a nonwatermarked layout. However, the inserted signatures can be easily erased by different watermark removal techniques in Section VI-1.

ICMarks: GW is resilient to forging attacks because the region with the minimal evaluation score is unique to the original nonwatermarked placement. The nonwatermarked layout and the scoring parameters are kept confidential. The ICMarks: GW signature verification requires the owner to provide that information to reproduce the watermark region R_w. Therefore, the adversary cannot counterfeit the watermarks without access to the nonwatermark layout. ICMarks: DW also exhibits similar properties, where the signatures

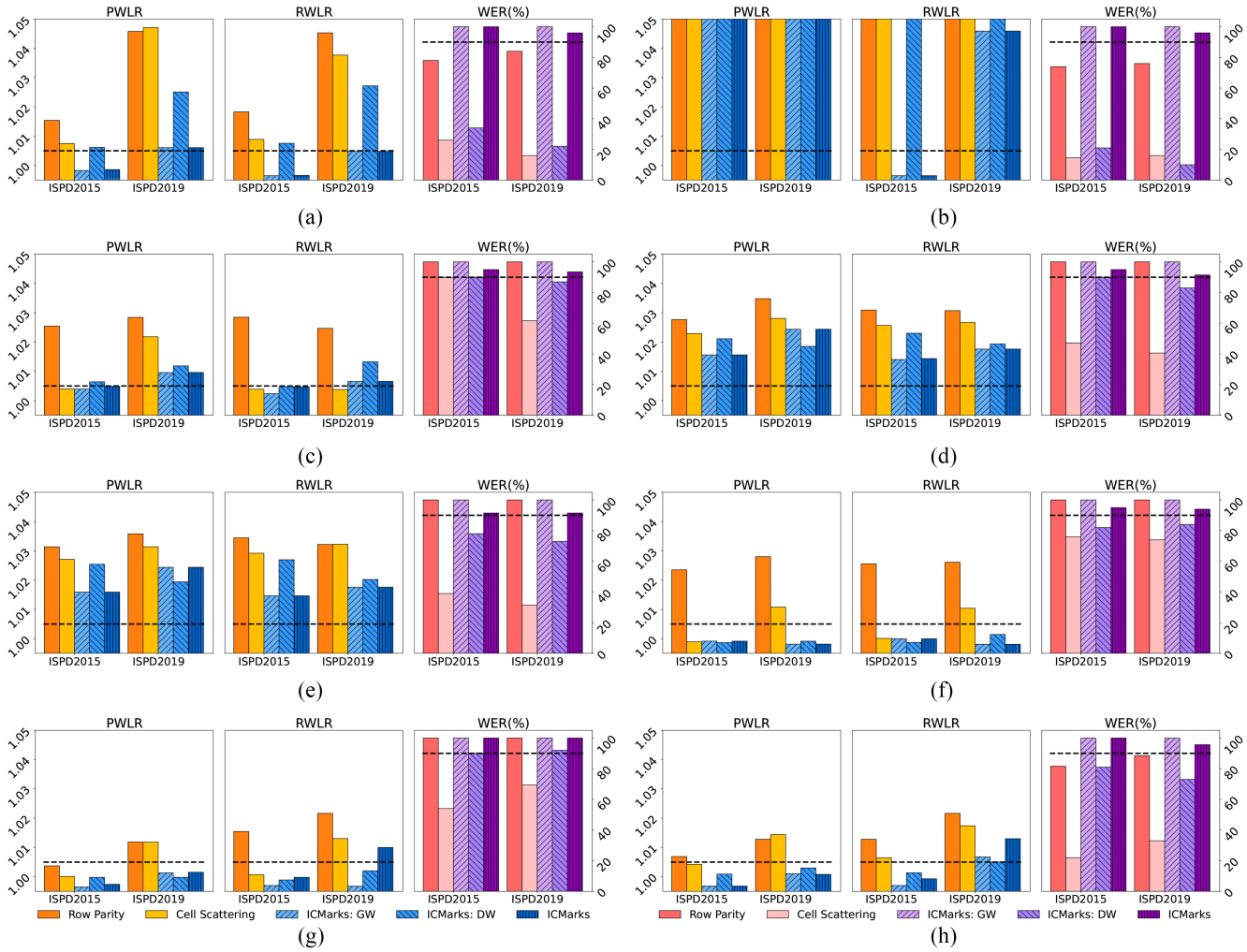


Fig. 10. WM performance under different attacks for wirelength-driven placement on the ISPD'2015 [27] and ISPD'2019 [28] benchmarks. The black dotted line in the two left subfigures denotes the quality degradation threshold of 1.005, and the black dotted line in the rightmost subfigure denotes the watermark extraction threshold of 90%. (a) SLA (0.1%). (b) SLA (0.5%). (c) CPA (0.1%). (d) CPA (1%). (e) CPA (10%). (f) OA. (g) ARA (top-1). (h) ARA (top-5).

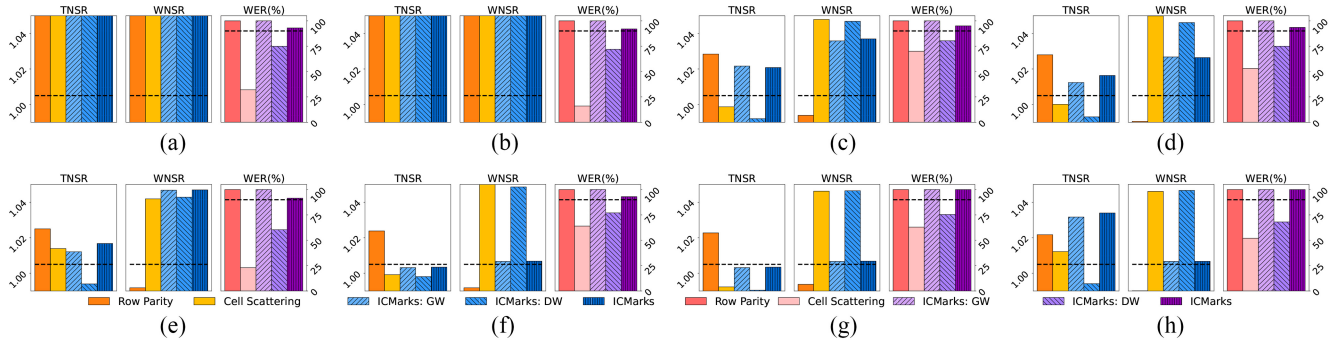


Fig. 11. WM performance under different attacks for timing-driven placement on the ICCAD'2015 benchmarks [29]. The black dotted line in the two left subfigures denotes the quality degradation threshold of 1.005, and the black dotted line in the rightmost subfigure denotes the watermark extraction threshold of 90%. (a) SLA (0.1%). (b) SLA (0.5%). (c) CPA (0.1%). (d) CPA (1%). (e) CPA (10%). (f) OA. (g) ARA (top-1). (h) ARA (top-5).

are encoded before detailed placement on the intermediate placement P_{itr}. As such, the adversary with only access to the watermarked layout cannot reproduce the watermarked cells or distance to forge the signature. ICMarks, as a combination of ICMarks: GW and ICMarks: DW is also resilient to watermark forging attacks.

We evaluate the probability that an adversary forges the ICMarks: GW signature, and we assume the adversary has prior knowledge of the scoring mechanism and watermarked region size. However, he does not have access to the scoring hyperparameters. Different from the ARAs in Watermark Removal Attacks, the adversary forges the signature by moving the cells in the searched region. We rescore the region using hyperparameters $\alpha = 25$, $\beta = 15$, and $\gamma = 10$.

We rank the scores the adversary gets from low to high, where lower scores indicate the region is more ideal for

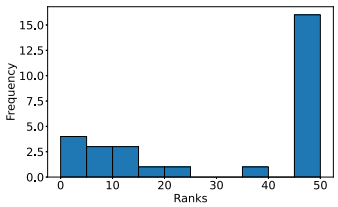


Fig. 12. Rank distribution of the watermarked region appears in the rescored watermarked region.

him/her to forge the signature. We show the rank of the watermarked region appears in the rescored watermarked layout in Fig. 12 on both ISPD'2019 [28] and ISPD'2015 [27] benchmarks. As seen, most of the layouts are ranked away from the top-5 regions, meaning the adversary cannot easily forge the watermarked region of cells.

VII. Conclusion

We present `ICMarks`, a robust WM framework for ICs physical design IP protection in the supply chain. We first introduce a GW method that identifies the watermarked region with insignificant performance degradation; then, we propose an independent DW technique to select cells that do not overlap with neighbors after perturbation to encode watermarks. Based on these methods, we develop `ICMarks`, which combines the best attributes of both GW and DW, thereby achieving minimal quality degradation with augmented robustness. Extensive experiments on ISPD'2015 [27], ISPD'2019 [28], and ICCAD'2015 [29] benchmarks demonstrate that `ICMarks` successfully inserts watermarks without compromising layout quality. Furthermore, we showcased `ICMarks`'s resiliency against watermark removal and forging attacks through comprehensive attack evaluations.

References

[1] B. Liu and G. Qu, "VLSI supply chain security risks and mitigation techniques: A survey," *Integration*, vol. 55, pp. 438–448, Sep. 2016.

[2] L. Pawar, R. Kumar, and A. Sharma, "Risks analysis and mitigation technique in EDA sector: VLSI supply chain," in *Analyzing the Role of Risk Mitigation and Monitoring in Software Development*. Hershey, PA, USA: IGI Glob., 2018, pp. 256–265.

[3] A. Cui, C.-H. Chang, and L. Zhang, "A hybrid watermarking scheme for sequential functions," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2011, pp. 2333–2336.

[4] D. Divyanshu, R. Kumar, D. Khan, S. Amara, and Y. Massoud, "FSM inspired unconventional hardware watermark using field-assisted SOT-MTJ," *IEEE Access*, vol. 11, pp. 8150–8158, 2023.

[5] X. Cai, Z. Gao, F. Bai, and Y. Xu, "A watermarking technique for hard IP protection in post-layout design level," in *Proc. 7th Int. Conf. ASIC*, 2007, pp. 1317–1320.

[6] G. Sun, Z. Gao, and Y. Xu, "A watermarking system for ip protection by buffer insertion technique," in *Proc. 7th Int. Symp. Qual. Electron. Design (ISQED)*, 2006, pp. 5–675.

[7] D. Saha and S. Sur-Kolay, "Watermarking in hard intellectual property for pre-fab and post-fab verification," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 5, pp. 801–809, May 2015.

[8] K. Shamsi, M. Li, K. Plaks, S. Fazzari, D. Z. Pan, and Y. Jin, "IP protection and supply chain security through logic obfuscation: A systematic overview," *ACM Trans. Design Autom. Electron. Syst.*, vol. 24, no. 6, pp. 1–36, 2019.

[9] J. Knechtel, S. Patnaik, and O. Sinanoglu, "Protect your chip design intellectual property: An overview," in *Proc. Int. Conf. Omni-Layer Intell. Syst.*, 2019, pp. 211–216.

[10] P. Slpsk, S. Ray, and S. Bhunia, "TREEHOUSE: A secure asset management infrastructure for protecting 3DIC designs," *IEEE Trans. Comput.*, vol. 72, no. 8, pp. 2306–2320, Aug. 2023.

[11] S. Sutardja, "1.2 The future of IC design innovation," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Tech. Dig.*, 2015, pp. 1–6.

[12] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014.

[13] M. Rathor and G. P. Rathor, "Hard-sign: A hardware watermarking scheme using dated handwritten signature," *IEEE Design Test*, vol. 41, no. 2, pp. 75–83, Apr. 2024.

[14] H. Chen, C. Fu, B. D. Rouhani, J. Zhao, and F. Koushanfar, "Intellectual property protection of deep-learning systems via hardware/software co-design," *IEEE Design Test*, vol. 41, no. 2, pp. 23–31, Apr. 2024.

[15] A. Tauhid, L. Xu, M. Rahman, and E. Tomai, "A survey on security analysis of machine learning-oriented hardware and software intellectual property," *High-Confid. Comput.*, vol. 3, Jun. 2023, Art. no. 100114.

[16] D. Saha, P. Dasgupta, S. Sur-Kolay, and S. Sen-Sarma, "A novel scheme for encoding and watermark embedding in VLSI physical design for IP protection," in *Proc. Int. Conf. Comput., Theory Appl. (ICCTA)*, 2007, pp. 111–116.

[17] F. Bai, Z. Gao, Y. Xu, and X. Cai, "A watermarking technique for hard IP protection in full-custom IC design," in *Proc. Int. Conf. Commun., Circuits Syst.*, 2007, pp. 1177–1180.

[18] M. Ni and Z. Gao, "Watermarking system for IC design IP protection," in *Proc. Int. Conf. Commun., Circuits Syst.*, 2004, pp. 1186–1190.

[19] A. B. Kahng et al., "Constraint-based watermarking techniques for design IP protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 10, pp. 1236–1252, Oct. 2001.

[20] A. B. Kahng et al., "Robust IP watermarking methodologies for physical design," in *Proc. 35th Annu. Design Autom. Conf.*, 1998, pp. 782–787.

[21] T. Nie, T. Kisaka, and M. Toyonaga, "A watermarking system for IP protection by a post layout incremental router," in *Proc. 42nd Annu. Design Autom. Conf.*, 2005, pp. 218–221.

[22] A. E. Dunlop and B. W. Kernighan, "A procedure for placement of standard-cell VLSI circuits," *IEEE Trans. Comput.-Aided Design*, vol. 4, no. 1, pp. 92–98, Jan. 1985.

[23] R.-S. Tsay and E. Kuh, "A unified approach to partitioning and placement (VLSI layout)," *IEEE Trans. Circuits Syst.*, vol. 38, no. 5, pp. 521–533, May 1991.

[24] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proc. IEEE*, vol. 102, no. 8, pp. 1207–1228, Aug. 2014.

[25] R. S. Rajarathnam, Y. Lin, Y. Jin, and D. Z. Pan, "ReGDS: A reverse engineering framework from GDSII to gate-level netlist," in *Proc. IEEE Int. Symp. Hardw. Oriented Security Trust (HOST)*, 2020, pp. 154–163.

[26] L. Alrahis et al., "GNN-RE: Graph neural networks for reverse engineering of gate-level netlists," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 8, pp. 2435–2448, Aug. 2022.

[27] I. S. Bustany, D. Chinnery, J. R. Shinnerl, and V. Yutsis, "ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement," in *Proc. Symp. Int. Symp. Phys. Design*, 2015, pp. 157–164.

[28] W.-H. Liu, S. Mantik, W.-K. Chow, Y. Ding, A. Farshidi, and G. Posser, "ISPD 2019 initial detailed routing contest and benchmark with advanced routing rules," in *Proc. Int. Symp. Phys. Design*, 2019, pp. 147–151.

[29] M.-C. Kim, J. Hu, J. Li, and N. Viswanathan, "ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2015, pp. 921–926.

[30] Y. Qiu, Y. Xing, X. Zheng, P. Gao, S. Cai, and X. Xiong, "Progress of placement optimization for accelerating VLSI physical design," *Electronics*, vol. 12, no. 2, p. 337, 2023.

[31] S. Pawanekar, G. Trivedi, and K. Kapoor, "A nonlinear analytical optimization method for standard cell placement of vlsi circuits," in *Proc. 28th Int. Conf. VLSI Design*, 2015, pp. 423–428.

[32] A. Agnesina, K. Chang, and S. K. Lim, "VLSI placement parameter optimization using deep reinforcement learning," in *Proc. IEEE/ACM 39th Int. Conf. Comput.-Aided Design*, 2020, pp. 1–9.

[33] J.-M. Lin, C.-W. Huang, L.-C. Zane, M.-C. Tsai, C.-L. Lin, and C.-F. Tsai, "Routability-driven global placer target on removing global and local congestion for VLSI designs," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design (ICCAD)*, 2021, pp. 1–8.

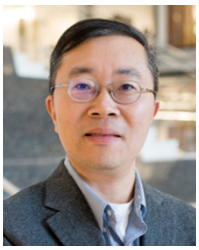
[34] Y. Lin, S. Dhar, W. Li, H. Ren, B. Khailany, and D. Z. Pan, "DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern vlsi placement," in *Proc. 56th Annu. Design Autom. Conf.*, 2019, pp. 1–6.
[35] P. Liao, S. Liu, Z. Chen, W. Lv, Y. Lin, and B. Yu, "DREAMPlace 4.0: Timing-driven global placement with momentum-based net weighting," in *Proc. Design, Autom. Test Europe Conf. Exhib. (DATE)*, 2022, pp. 939–944.
[36] P. Spindler, U. Schlichtmann, and F. M. Johannes, "Abacus: Fast legalization of standard cell circuits with minimal movement," in *Proc. Int. Symp. Phys. Design*, 2008, pp. 47–53.
[37] R. Netto et al., "Algorithm selection framework for legalization using deep convolutional neural networks and transfer learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 5, pp. 1481–1494, May 2022.
[38] H. Yang, K. Fung, Y. Zhao, Y. Lin, and B. Yu, "Mixed-cell-height legalization on CPU-GPU heterogeneous systems," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2022, pp. 784–789.
[39] M. Pan, N. Viswanathan, and C. Chu, "An efficient and effective detailed placement algorithm," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2005, pp. 48–55.
[40] W.-K. Chow, J. Kuang, X. He, W. Cai, and E. F. Y. Young, "Cell density-driven detailed placement with displacement constraint," in *Proc. Int. Symp. Phys. Design*, 2014, pp. 3–10.
[41] G. T. Becker, M. Kasper, A. Moradi, and C. Paar, "Side-channel based watermarks for integrated circuits," in *Proc. IEEE Int. Symp. Hardw.-Oriented Security Trust (HOST)*, 2010, pp. 30–35.
[42] U. Das et al., "PSC-watermark: Power side channel based IP watermarking using clock gates," in *Proc. IEEE Eur. Test Symp. (ETS)*, 2023, pp. 1–6.
[43] M. Shayan, K. Basu, and R. Karri, "Hardware trojans inspired IP watermarks," *IEEE Design Test*, vol. 36, no. 6, pp. 72–79, Dec. 2019.
[44] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly stripping functionality for logic locking: A fault-based perspective," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4439–4452, Dec. 2020.
[45] H. M. Kamali et al., "Advances in logic locking: Past, present, and prospects," Cryptol. ePrint Arch., IACR, Bellevue, WA, USA, Rep. 2022/260, 2022.
[46] P. Santikellur, R. S. Chakraborty, and S. Bhunia, "Hardware IP protection using register transfer level locking and obfuscation of control and data flow," in *Behavioral Synthesis for Hardware Security*. Cham, Switzerland: Springer Int. Publ., 2021, pp. 57–69.
[47] P. Bagul and V. Inamdar, "Hardware obfuscation based watermarking technique for IPR ownership identification," *Int. J. Reconfigurable Comput.*, vol. 1, no. 1, 2023, Art. no. 4550758.
[48] J. Gu, Z. Jiang, Y. Lin, and D. Z. Pan, "DREAMPlace 3.0: Multi-electrostatics based robust VLSI placement with region constraints," in *Proc. IEEE/ACM 39th Int. Conf. Comput.-Aided Design*, 2020, pp. 1–9.
[49] J. Lu et al., "ePlace: Electrostatics-based placement using fast fourier transform and Nesterov's method," *ACM Trans. Design Autom. Electron. Syst.*, vol. 20, no. 2, pp. 1–34, 2015.
[50] C.-K. Cheng, A. B. Kahng, I. Kang, and L. Wang, "RePlAce: Advancing solution quality and routability validation in global placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 9, pp. 1717–1730, Sep. 2019.
[51] J. Liu, C.-W. Pui, F. Wang, and E. F. Y. Young, "CUGR: Detailed-Routability-driven 3D global routing with probabilistic resource model," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, 2020, pp. 1–6.
[52] T.-W. Huang, G. Guo, C.-X. Lin, and M. D. F. Wong, "OpenTimer v2: A new parallel incremental timing analysis engine," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 4, pp. 776–789, Apr. 2021.
[53] D. Saha and S. Sur-Kolay, "Fast robust intellectual property protection for VLSI physical design," in *Proc. 10th Int. Conf. Inf. Technol. (ICIT)*, 2007, pp. 1–6.
[54] W. Liang, X. Sun, Z. Xia, D. Sun, and J. Long, "A chaotic IP watermarking in physical layout level based on FPGA," *Radioengineering*, vol. 20, no. 1, pp. 118–125, 2011.
[55] D. R. Anderson, K. P. Burnham, and W. L. Thompson, "Null hypothesis testing: Problems, prevalence, and an alternative," *J. Wildlife Manage.*, vol. 64, no. 4, pp. 912–923, Oct. 2000.
[56] A. B. Kahng and T. Spyrou, "The OpenROAD project: Unleashing hardware innovation," in *Proc. GOMAC*, 2021, pp. 1–6.

Ruisi Zhang received the B.E. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2021. She is currently pursuing the Ph.D. degree advised by Prof. F. Koushanfar with the Department of Electrical and Computer Engineering, University of California at San Diego, San Diego, CA, USA.

Her research interests include machine learning security and acceleration.

Rachel Selina Rajarathnam (Senior Member, IEEE) received the B.E. degree in electronics and communication engineering and the M.E. degree in applied electronics from Anna University, Chennai, India, in 2011 and 2013, respectively, and the Ph.D. degree in electrical and computer engineering from University of Texas at Austin, Austin, TX, USA, under the supervision of Prof. D. Z. Pan.

Her current research interests include physical design automation and hardware security.



David Z. Pan (Fellow, IEEE) received the B.S. degree from Peking University, Beijing, China, and the M.S. and Ph.D. degrees from University of California, Los Angeles, CA, USA.

He is the Silicon Laboratories Endowed Chair Professor with the ECE Department, The University of Texas at Austin, Austin, TX, USA. He has graduated over 50 Ph.D. students and postdocs who are now holding key academic and industry positions. He has published over 500 refereed journal/conference papers and nine U.S. patents. His research interests include electronic design automation, synergistic AI and IC co-optimizations, design for manufacturing, and CAD for analog/mixed-signal/RF and emerging technologies.

Dr. Pan has received many awards, including 21 Best Paper Awards from premier venues, the SRC Technical Excellence Award, the DAC Top 10 Author Award in Fifth Decade, and the ASP-DAC Frequently Cited Author Award. He has served on many editorial boards and conference committees, e.g., as the DAC 2024 TPC Chair and the ICCAD 2019/2018 General/TPC Chair. He is a Fellow of ACM and SPIE.

Farinaz Koushanfar (Fellow, IEEE) received the Ph.D. degree from University of California, Berkeley, CA, USA, in 2005.

She is the Siavouche Nemat-Nasser Endowed Chair Professor of ECE with the Jacobs School of Engineering, University of California at San Diego, San Diego, CA, USA. She has published over 250 refereed journal/conference papers in premier venues and holds 25 patents. Her current research addresses several aspects of secure and efficient computing, with a focus on AI-based co-optimization, robust machine learning under resource constraints, hardware and system security, intellectual property protection, design automation, as well as cryptographically secure privacy-preserving computing.

Dr. Koushanfar has received a number of awards and honors, including the Presidential Early Career Award for Scientists and Engineers (PECASE) from President Obama, the ACM SIGDA Outstanding New Faculty Award, the Cisco IoT Security Grand Challenge Award, the MIT Technology Review TR-35, the Qualcomm Innovation Awards, the Intel Collaborative Awards, the Young Faculty/CAREER Awards from NSF, DARPA, ONR, and ARO, as well as several best paper awards. She has been serving on multiple editorial boards and conference organizations, such as the editorial board of the PROCEEDINGS OF THE IEEE, the NDSS'22 TPC Chair, and the WiSEC'24 TPC Chair. She is a Fellow of ACM, National Academy of Inventors, and the Kavli Frontiers of the National Academy of Sciences.