GRAIN: A Design-Intent-Driven Analog Layout Migration Framework

Bingyang Liu*†, Haoning Jiang‡, Haoyi Zhang*, Xiaohan Gao*†, Zichen Kong*,
Xiyuan Tang*, David Z. Pan†, Yibo Lin*
*Peking University, Beijing, China †UT Austin, Austin, TX, USA ‡SUSTech, Shenzhen, China

***Abstract*—Migrating validated analog layouts from one technology to the next is an important problem, but it remains a labor-intensive task. Existing automatic layout migration methods often suffer from performance degradation as they fail to preserve the multi-level design intent embedded in manual layouts and the wealth of design knowledge they embody. Moreover, these methods often encounter issues such as Layout-Versus-Schematic (LVS) violations and instability during the migration process. In this work, we propose GRAIN: a design-intent-driven analog layout migration framework that applies a constraint-aware hierarchical placement migration for multi-level placement behavior migration, and performs guide-based routing that decouples similarity from legality via a maze router to achieve LVS-clean results. The experiment in real analog designs migrated from 65nm to 40nm and 28nm shows that, compared to the most recent analog layout migration framework to the best of our knowledge, GRAIN delivers 100% LVS-clean layouts without manual fixes and reduces area by 13.8% and wirelength by 29.2% on average, while also yielding post-layout metrics closer to the schematic.**

I. Introduction

The continuous scaling of semiconductor manufacturing processes has amplified the demand for efficient analog layout migration. Unlike layout synthesis, which generates completely new layout from a netlist, migration aims to transfer an existing, validated layout to a new technology node while preserving its performance and, crucially, the original design intent. This task is particularly challenging for analog circuits due to their high sensitivity to device matching, parasitics, and layout-dependent effects. Naive geometric scaling often fails because well-crafted analog layouts deliberately mitigate parasitics and layout-dependent effects, and if that intent is not preserved in the migrated layout, the circuit performance can degrade substantially. Consequently, industrial layout migration remains a largely manual, labor-intensive process, demanding experienced designers to meticulously adapt highly customized layouts.

Prior research has advanced automated analog layout migration, but shared limitations still hinder the practical adoption. Early methods focused on template-based migration [1], [2], which failed to capture the full behavioral patterns of routing [3]. More recent efforts have widely adopted Constraint-Graph (CG) [3]–[6] and topology slicing tree [7], [8] for placement migration. As for routing migration, heuristic methods including Constrained Delaunay Triangulation (CDT) [4] and Cartesian Detection Lines (CDL) [3], [5] have also gained traction. These approaches, however, suffer from three fundamental problems. First, they fail to preserve design intent in a holistic manner. A well-crafted analog layout embodies a complex set of trade-offs and designer experience, including constraints for device positions, guard ring and n-well strategies, and specific routing behaviors. Existing tools, by focusing on isolated geometric or topological features, capture only local design intent instead of this multi-level intent, leading to performance degradation. Second, CGs are highly dependent on the orders of graph construction, making results unpredictable and ill-suited for iterative design; moreover, CGs encode relative-position relations among adjacent elements, which is a local feature of layout, while high-quality migration requires global, long-range guidance on device distribution that CGs cannot encode, including symmetry, alignment and well sharing. Third, routing migration frequently fails in LVS check, necessitating manual fixes that undermine the goal of

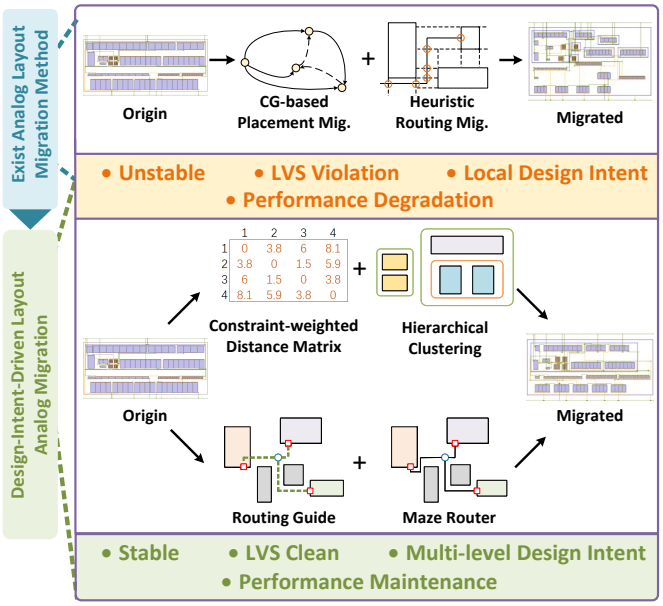

Fig. 1. A conceptual comparison of migration methodologies. Existing methods rely on unstable CG-based placement migration and heuristic routing migration algorithms, often resulting in LVS violations and performance degradation. Our design-intent-driven approach employs stable constraint-aware hierarchical clustering for placement migration and a guide-based, LVS-clean routing strategy, both enabling systematic preservation of multi-level design intent.

B.Liu and X.Gao was with Peking University when this work was conducted. They are now with the University of Texas at Austin.
Corresponding authors: Yibo Lin (yibolin@pku.edu.cn)

reducing human effort, which is due to an unsolved conflict in routing migration between source-layout similarity and LVS correctness in prior work.

To overcome these limitations, we draw methodological insights from automatic analog layout generation tools. While generation and migration address different tasks, the ideas related to constraint handling and routing abstraction remain relevant. For the LVS-clean and stability issues, we learn from the mature layout generation solution in MAGICAL [9]–[11], ALIGN [12]–[14], and recent works [15]–[17]. For the multi-level design-intent representation, we learn from code-based tools [18]–[22] and introduce an intermediate representation to capture and express high-level design intent.

In this paper, we propose GRAIN: a design-intent-driven layout migration framework, including a constraint-aware hierarchical clustering algorithm and a guide-based routing algorithm to resolve the conflict between similarity and LVS correctness. To enable the acquisition and retention of multi-level design intents, the framework maintains a progressively built intermediate representation in the migration process. Our contributions are summarized as follows:

- We develop a stable, design-intent-aware constraint-aware hierarchical clustering algorithm. Powered by DBSCAN algorithm, this method incorporates intra-cluster and inter-cluster compacting, balancing layout compactness while retaining multi-level key design features.
- We present a guide-based routing algorithm that resolves the long-standing conflict between routing similarity and LVS correctness. This approach first migrates an abstracted routing guide to preserve topological similarity, and then employs a maze router that adheres to this guide while ensuring an LVS-clean routing result.
- The experiment in real analog designs migrated from 65 nm to 40nm and 28nm shows that GRAIN delivers 100% LVS-clean layouts without manual fixes and reduces area by 13.8% and wirelength by 29.2% on average, compared to the most recent analog layout migration framework to the best of our knowledge. GRAIN further yields post-layout performance closer to the schematic than the same baseline, with the most noticeable improvement at 28nm.

The rest of the paper is organized as follows. Section II describes the background; Section III explains the detailed implementation; Section IV demonstrates the results; Section V concludes the paper.

II. Preliminaries

This section reviews works for analog layout migration, focusing on placement migration and routing migration. Additionally, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is briefly introduced.

A. Analog Placement Migration

Analog placement migration aims to preserve the layout's spatial structure during technology transitions. Early methods relied on symbolic templates to maintain topological regularity. For instance, [23] proposed hierarchical symmetry detection and reconstruction using parameterized templates. While effective in preserving geometric patterns, such approaches required manual tuning for device scaling and lacked flexibility across technology variations. Later, [24] combined sizing migration and 1-d constraint-graph driven layout migrarion, and [25] introduced a constraint aware multilevel clustering method for placement migration. Symbolic structures were further combined with mathematical optimization: [26] applied geometric programming for constraint-aware migration, while [27] incorporated parasitic models into the migration process. Despite these advances, template-based methods remain limited in scalability and fail to generalize across diverse designs [3].

To address these issues, more mature constraint-graph (CG) based layout migration methods were introduced [28]–[30]. [3] used Sequence Pair (SP) representations [31], [32] to generate Horizontal and Vertical Constraint Graphs (HCG/VCG) via depth-first search. These directed acyclic graphs capture relative device positions and allow multiple candidate SPs, evaluated using routability-aware cost functions. This enables area-focused placement migrations under non-overlap constraints, and the introduction of analytical method also makes migration more adaptable to different designs. However, CG-based methods is inherently unstable and hard to control. The migration result can vary largely depending on the order in which the graph is constructed. Additionally, this method operates at a coarse granularity, failing to preserve the specific design intents such as symmetry, alignment, and exact spacing between critical devices. These drawbacks make CG-based approaches not good enough for analog layout migration tasks.

B. Analog Routing Migration

The goal of routing migration is to generate an LVS-clean layout in the target technology while preserving the original routing characteristics. Due to the complexity of routing migration, early works [23], [24], [26], [27], [33] on layout migration primarily focused on placement, treating routing as a post-processing step or leaving it to manual effort. [4] first proposed CDT method, using triangle-based decomposition to capture geometric correlations between routing paths and placement blocks. However, CDT is sensitive to device size and spacing changes, and can lead to unnecessary detours in routing. To address this, [3] proposed the CDL model, which uses adaptive orthogonal detection lines to build a better representation and reduce disconnections. While an improvement, both CDT and CDL are limited to planar topology and lack sufficient adaptability to multi-layer structures, leading to LVS violations when migrating complex cross-layer routing behavior, especially in routing-dense or near-pin regions. These limitations motivate the need for a routing migration approach that ensures LVS correctness and better preserves design intent.

C. DBSCAN Algorithm

DBSCAN [34] is a density-based clustering algorithm that groups points based on local density. It relies on two parameters: ϵ, the neighborhood radius, and MinPts, the minimum number of points required to form a dense region. A point is classified as a core point if it has at least MinPts neighbors

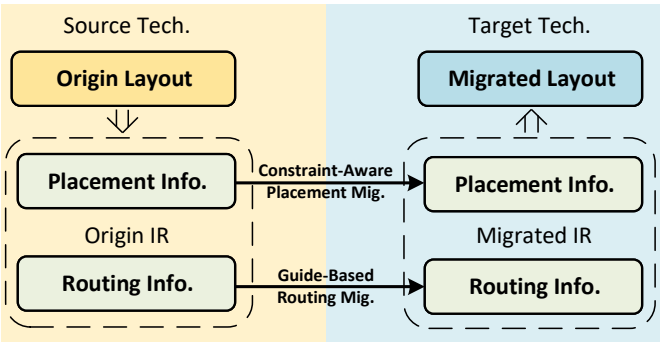

Fig. 2. Overview of the GRAIN framework. Our algorithms extract multi-level design intent from the source layout, progressively building an intermediate representation (IR). This abstracted intent is then migrated to the target technology, and the final layout is reconstructed from the migrated IR.

within its ϵ-radius. Points within the ϵ-radius of a core point but with fewer neighbors are called border points. Points that are not density-reachable from any core point are labeled as noise. The reachability condition between two points p and q is satisfied if:

$$\text{Dist}(p,q) \leq \epsilon \quad \text{and} \quad |\{r : \text{Dist}(r,q) \leq \epsilon\}| \geq \text{MinPts}. \quad (1)$$

Unlike clustering methods such as K-Means or hierarchical clustering, which assume compact or spherical clusters, DBSCAN can detect clusters of arbitrary shapes. As for analog layout, device groups divided by guard rings and nwells often form elongated or irregular structures. As such, DBSCAN is well-suited to our placement migration framework for the clustering task.

III. GRAIN Framework

The goal of GRAIN framework is to migrate an existing analog layout to a target technology while preserving multi-level design intent and delivering LVS-clean migration results. The raw polygonal layout in GDSII format is too fine-grained; multi-level design intent is implicitly embedded within a sea of polygon coordinates, making it difficult to manipulate directly for migration. We therefore adopt a unified strategy for both placement and routing, as shown in 2: (1) extract multi-level intent from the source layout into an explicit intermediate representation (IR); (2) migrate this abstracted intent to the target technology; and (3) reconstruct the target layout from the migrated IR. The IR is built incrementally by our algorithms and serves as the bridge that carries intent across stages. For placement, the IR captures critical constraints to minimize mismatch, along with clustering results that maintains well sharing and guard ring behavior. For routing, it contains normalized net-level information including routing paths, pin and junction positions, and the relative position of each pin to its device.

A. Constraint-Aware Hierarchical Placement Migration

To systematically capture and transfer multi-level design intent, our placement migration employs a constraint-aware and hierarchical strategy. This methodology is designed to overcome the instability inherent in traditional CG-based methods while systematically preserving complex design intent, including symmetry and alignment constraints of critical devices, n-well sharing and guard ring behavior. As illustrated in Figure 3, the process involves four main steps: (a) initializing device positions, (b) constraint-aware clustering, (c) intra-cluster compaction, and (d) inter-cluster compaction.

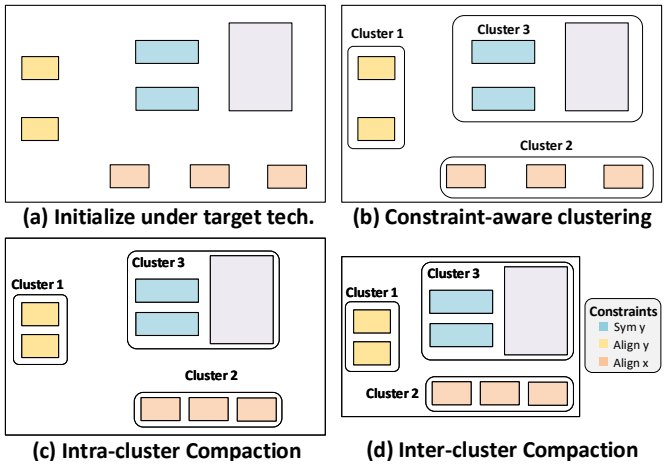

Fig. 3. The constraint-aware hierarchical placement migration process, consisting of four major steps: (a) initializing the device positions under the target technology, (b) performing constraint-aware clustering, (c) intra-cluster compaction, and (d) inter-cluster compaction.

a) Initial Migration: We begin by scaling the device center positions from the source layout to the target technology, preserving the original aspect ratio. This ensures that symmetry and alignment constraints are maintained, while preventing overlaps. The result is a rough but legal initial placement that reflects the structure of the original layout, albeit with sparse distribution.

b) Constraint-Aware Clustering: To preserve spatial patterns such as dense device groups and reserved guard ring regions, we apply the DBSCAN algorithm. Each constraint S_j is associated with a strength coefficient $\lambda_j \in [0,1]$, and the constraint matrix $S = \{s_{ij}\}$ is defined such that $s_{ij} = \lambda_j$ for devices sharing S_j. We compute a distance matrix combining spatial proximity and constraint strength:

$$d_{ij} = \min_{p_1,p_2} \text{Dist}(p_1,p_2) \cdot s_{ij}, \quad p_{1,2} \in BBox_{i,j}, \quad (2)$$

where $BBox_i$ is the bounding box of device i, p_1, p_2 are points, and function $Dist(\cdot)$ measures the distance from point to point in a plane. Devices are then clustered in this transformed space, ensuring those with strong mutual constraints are grouped together.

c) Intra-Cluster Compaction: Within each cluster, devices are compacted using a greedy strategy that iteratively moves them toward the cluster center. The movement vector for a device D_i is computed as:

$$\vec{\delta_i} = \vec{v}_{ic} \cdot \frac{t_{std}}{\min_{p_1} \text{Dist}(p_1,p_c)}, \quad p_1 \in BBox_i, \quad (3)$$

where t_{std} is a technology-specific step size and $\vec{v}_{ic}$ is the vector from the device center to the cluster center p_c. Devices involved in symmetry or alignment constraints move in a coordinated manner. If spacing violations occur, the move is rolled back and retried along restricted axes.

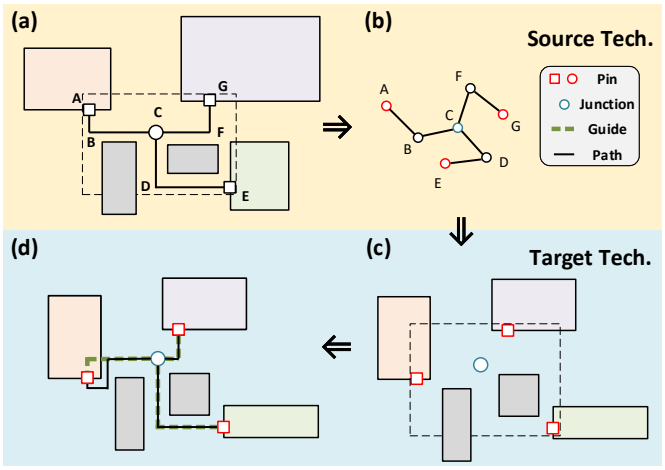

Fig. 4. Guide-based routing migration process, consisting of four major steps: (a) extraction of the routing guide, (b) abstraction of the routing path into a routing guide, with pin and junction information extracted, (c) reconstruction of pins and junctions in the target technology, and (d) reconstruction of the routing guide, followed by completing the routing migration with a maze router.

d) Inter-Cluster Compaction: After intra-cluster compaction, each cluster is treated as a higher-level rectangle, and then the rectangles are compacted similarly. This hierarchical process is then applied recursively: these macro-blocks are themselves clustered and compacted. This iterative clustering preserves the high-level spatial distribution of the original layout, which is critical for ensuring that devices originally sharing an n-well or enclosed by the same guard ring remain grouped together after migration. The process continues until all elements merge into a single, compact region, yielding a final migrated placement that balances design intent preserving and spatial efficiency.

B. Guide-based Routing Migration

To capture and transfer routing design intent at the net level, rather than relying solely on polygon geometries, our routing migration introduces a guide-based approach. This enables LVS-clean routing reconstruction in the target technology while preserving the routing features of the original layout. As illustrated in Figure 4, the flow includes guide extraction and decomposition in the source technology (a)(b), followed by guide migration (c) and detailed routing in the target technology (d).

a) Guide Extraction and Abstraction: We represent the routing space as a 3D grid, where each routing path is abstracted as a collection of wires forming a net $N_i = \{w_{ij}\}$. Each wire w_{ij} connects two adjacent grid points p_{ij1}, p_{ij2}, satisfying exactly one of x, y, z differs between p_{ij1} and p_{ij2}, so that each wire spans one grid edge. The net N_i forms an undirected, acyclic graph where edges are wires and nodes are connection points (Figure 4(b)).

We further extract two structural elements from each net: Pin P_i means leaf node in the graph (degree = 1), each tied to a device D_{ik}; Junction C_i means high-degree node (degree ≥ 3). For each pin $p_{ik} \in P_i$, we record its position relative to the bounding box $\text{BBox}_{D_{ik}}$ of its associated device, enabling consistent pin reconstruction after placement migration.

b) Net Decomposition: To simplify migration, each net N_i is decomposed into a set of 2-pin nets $N_{it} = \{q_{it1}, q_{it2}\}$, where q_{it1} and q_{it2} are either pins or junctions. This is done by splitting the net at junction nodes, and tracing branches to adjacent pins or other junctions, preserving intermediate paths.

c) Guide Migration: In this step, we reconstruct the positions of pins and junctions as follows: Each pin p'_{ik} is placed at the same relative position inside the migrated $\text{BBox}_{D_{ik}}$; Each junction c'_{il} is placed at the same relative coordinate within the new bounding box formed by its adjacent pins, as shown in Fig 4 (c).

For each 2-pin net, we apply an axis-aligned affine transformation to map it from the source to the target layout. Let q_{it1} and q_{it2} be the endpoints of a 2-pin net in the source layout, and q'_{it1} and q'_{it2} be the corresponding endpoints in the target layout. We define the transformation matrix T as:

$$T = \begin{bmatrix} s_x & 0 & 0 & t_x \\ 0 & s_y & 0 & t_y \\ 0 & 0 & s_z & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where s_x, s_y, and s_z are scaling factors along the x, y, and z axes, and t_x, t_y, t_z are translation offsets. The matrix T is determined such that q_{it1} is mapped to q'_{it1} and q_{it2} is mapped to q'_{it2}. Each point p on the original 2-pin net is transformed as $p' = T \cdot [p_x, p_y, p_z, 1]^T$. Since the transformation is axis-aligned, it preserves the orientation of wire segments: segments that were aligned with coordinate axes remain so after migration. The set of all transformed 2-pin nets constitutes the migrated routing guide, as shown in Figure 4 (d).

d) Routing Path Reconstruction: We then apply a maze router to reconstruct LVS-clean paths, using the migrated guide as a soft constraint. Inspired by [16], [17], a penalty term is added to the cost function to encourage the actual routing path to align with the guide:

$$\text{Cost}(p_i) = \text{BaseCost}(p_i) + \lambda \cdot \text{Distance}(p_i, P_{\text{guide}}), \tag{5}$$

where p_i is a routing point, P_{guide} is the nearest guide path, and λ is a weight factor. According to [16], [17], the BaseCost contains components including obstacle penalty, wire length, and other constraints. This cost function biases the routing path toward the migrated guide while ensuring LVS correctness.

IV. Experimental Results

A. Experiment Setup

We evaluate our migration framework on real-world analog design migrated from 65nm to 40nm and 28nm technologies, and compare it with the most recent analog layout migration method to the best of our knowledge [3]. This work is selected as a representative baseline, as it improves upon earlier approaches such as [4], and closely resembles [5] in methodology, benchmarks, and experiments. Additionally, fully automated layout generation tools [9]–[17] are not included in our comparison, as they generate layouts from scratch and are not designed for intent-preserving migration across technologies.

TABLE I
OTA: GEOMETRIC METRICS AND POST-LAYOUT PERFORMANCE

Tech.	Method	Area (um^2)	Const.	WL (um)	LVS	Gain (dB) [ΔdB]	-3dB (kHz) [Δ%]	CMRR (dB)	PSRR (dB)	PM (deg) [Δ°]
65nm	Sch.	N/A	N/A	N/A	N/A	70.45	63.24	$>$120	$>$120	69.32
	Manual	2001.1	7/7	523.5	✓	62.78 (-7.67)	34.79 (-45.0%)	85.93	80.77	81.84 (+12.52)
40nm	Sch.	N/A	N/A	N/A	N/A	55.79	109.6	$>$120	$>$120	77.74
	[3]	1391.8	5/7	893.2	✗	-	-	-	-	-
	[3] fix	1391.8	5/7	697.9	✓	53.75 (-2.04)	56.90 (-48.1%)	62.64	77.22	**81.18 (+3.44)**
	[3]+Ours	1391.8	5/7	566.8	✓	54.94 (-0.85)	58.90 (-48.1%)	62.07	71.62	86.81 (+9.07)
	Ours	**1383.9**	**7/7**	**521.3**	✓	**55.48 (-0.31)**	**67.10 (-38.8%)**	**63.09**	**74.86**	86.66 (+8.62)
28nm	Sch.	N/A	N/A	N/A	N/A	69.54	273.3	$>$120	$>$120	68.88
	[3]	520.2	2/7	572.9	✗	-	-	-	-	-
	[3] fix	520.2	2/7	434.4	✓	27.11 (-42.13)	1112 (+307%)	38.41	26.79	50.90 (-17.98)
	[3]+Ours	520.2	2/7	402.0	✓	41.20 (-28.34)	**1318 (+383%)**	38.07	35.04	31.69 (-37.19)
	Ours	**362.2**	**7/7**	**305.7**	✓	**58.75 (-10.79)**	224.6 (+17.8%)	**44.08**	**38.92**	**78.91 (+9.31)**

Notes: N/A means not applicable to schematic; - means unavailable due to LVS check failure; $>$120 dB means CMRR/PSRR reaching the upper limit of simulation accuracy. To facilitate comparison between layout and schematic performances under the same technologies, deltas are shown in parentheses.

TABLE II
COMP: GEOMETRIC METRICS AND POST-LAYOUT PERFORMANCE

Tech.	Method	Area (um^2)	Const.	WL (um)	LVS	t_r (ns) [Δns]	t_f (ns) [Δns]	V_{os}/V_{FS} (%) [Δpp]	V_{hys}/V_{FS} (%) [Δpp]
65nm	Sch.	N/A	N/A	N/A	N/A	19.8	8.25	0.92	5.48
	Manual	262.3	3/3	124.78	✓	20.3 (+0.5)	10.1 (+1.85)	1.49 (+0.57)	5.65 (+0.17)
40nm	Sch.	N/A	N/A	N/A	N/A	16.0	8.11	1.02	5.96
	[3]	166.0	2/3	260.6	✗	-	-	-	-
	[3] fix	166.0	2/3	208.4	✓	**18.5 (+2.5)**	9.77 (+1.66)	1.65 (+0.63)	7.93 (+1.97)
	[3]+Ours	166.0	2/3	154.3	✓	18.6 (+2.6)	9.54 (+1.43)	1.65 (+0.63)	8.19 (+2.23)
	Ours	**136.0**	**3/3**	**146.6**	✓	**18.5 (+2.5)**	**9.23 (+1.12)**	**1.60 (+0.58)**	**6.80 (+0.84)**
28nm	Sch.	N/A	N/A	N/A	N/A	3.99	2.17	0.14	5.60
	[3]	115.6	2/3	182.4	✗	-	-	-	-
	[3] fix	115.6	2/3	166.9	✓	4.77 (+0.78)	2.91 (+0.74)	0.60 (+0.46)	4.24 (+1.36)
	[3]+Ours	115.6	2/3	128.7	✓	4.58 (+0.59)	2.87 (+0.70)	0.39 (+0.25)	3.45 (+2.15)
	Ours	**77.14**	**3/3**	**107.8**	✓	**4.42 (+0.43)**	**2.80 (+0.63)**	**0.38 (+0.24)**	**4.25 (+1.35)**

Notes: See shared notes under Table I.

Therefore, we employed the router proposed by [16], [17] as the routing engine in our framework.

We consider two manually-crafted 65nm designs with distinct functions, topologies, and design intents: a two-stage folded-cascode OTA and a comparator (COMP). Each is migrated to 40nm and 28nm. Post-layout simulation is performed in Cadence Virtuoso with parasitic extraction and LVS checking via Mentor Graphics Calibre.

A challenge for a fair comparison is that the routing migration algorithm of baseline [3] fails to produce LVS-clean layouts. To enable a meaningful post-layout analysis, we designed a set of hybrid baselines: [3] fix: Uses the baseline's placement, with its non-LVS-clean routing output serving as a guide for our maze router to generate a legal layout. [3] + Ours: Uses the baseline's placement but with our complete guide-based routing algorithm. This experimental design allows us to isolate the contributions of different components: comparing [3] vs. the two mixed variants isolates routing effects; comparing [3]+Ours vs. Ours isolates placement effects. All four migration outcomes are further contrasted with the schematic to assess how well design intent is preserved within each technology. Notably, [3]'s placement migration yielded hundreds of different results due to instability. We selected the result with the smallest displacement from the original layout for subsequent post-layout simulation comparison. All circuit migrations are completed within minutes. A visualization of the OTA migration is provided in Fig. 5, while COMP is not shown due to space limitations.

B. Results and Analysis

The geometric and post-layout performance metrics for the OTA and COMP are detailed in Tables I and II, respectively. Geometric metrics include layout area (Area), preservation of constraints (Const.), and total wire length (WL). The post-layout performance metrics differ for each circuit. For the OTA, we measure loop gain (Gain), -3dB bandwidth, common-mode rejection ratio (CMRR), power supply rejection ratio (PSRR), and phase margin (PM). For these metrics, higher values are generally better, except for PM, which indicates circuit stability and should ideally remain between 60-80° and close to the schematic. For the COMP, we evaluate rise/fall times (t_r, t_f), normalized offset voltage (V_{os}/V_{FS}),

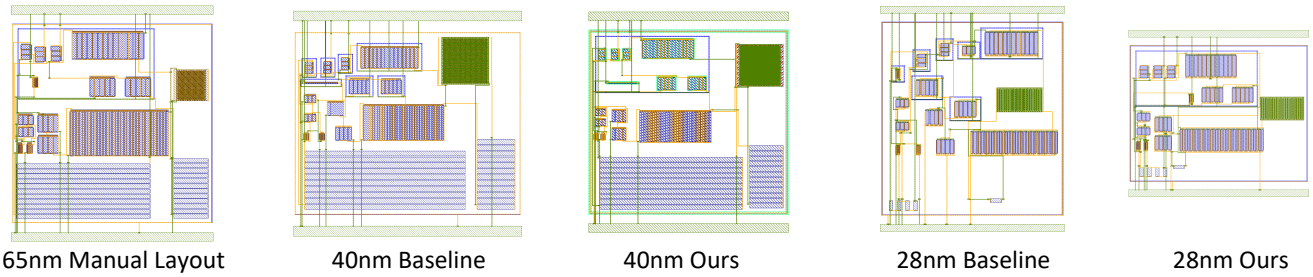

Fig. 5. Visual comparison of the migrated OTA layouts. "Baseline" refers to the result using the placement from [3] and our routing-fix, while "Ours" is the result from our complete framework. Our method consistently produces more compact layouts that better preserve the structural arrangement and symmetry of the original 65nm manual layout across both 40nm and 28nm technologies.

and normalized hysteresis voltage (V_{hys}/V_{FS}). Lower delay and offset are desirable, while hysteresis should remain close to the schematic's value.

To properly evaluate the quality of a layout migration, it is necessary to note that comparisons of absolute performance metrics across different technologies can be misleading due to sizing and process variations. The most salient measure of success is the performance degradation from the schematic to the post-layout simulation within the same technology node. This delta quantifies how well a layout preserves the intended circuit behavior against parasitic effects, which become increasingly dominant as technology scales. We also note that analog performance often involves trade-offs; therefore, we assess the overall similarity to the schematic's behavior rather than focusing on a single metric.

The most immediate result, evident in both tables, is that the baseline [3] fails LVS checks in all test cases, rendering its output unusable for simulation without manual repair. In contrast, all migration results from GRAIN and the hybrid methods are LVS-clean, demonstrating the advantage on practicality of our guide-based routing migration approach.

By comparing [3] fix and [3]+Ours, we can isolate routing migration methods. Across both designs and both target technologies, our guide generation yields consistently better overall post-layout performance and noticeably shorter wire length. Gains are bounded, however, by the inherited placement, which limits headroom for performance recovery. For example, in the 40nm OTA migration, our routing guide leads to a 19% reduction in WL and cuts the gain degradation by more than half (from -2.04 dB to -0.85 dB) compared to using the baseline's output as a guide, while maintaining other metrics almost the same. This shows that our abstracted routing guide preserves the original routing intent more effectively.

Comparing [3]+Ours with Ours highlights the difference between the placement migration methods. Replacing the baseline placement with our constraint-aware hierarchical placement reduces area and improves constraint satisfaction, while further improving overall post-layout performance. The most telling results are in the metrics highly sensitive to parasitic matching, including CMRR, PSRR, and V_{os}/V_{FS}. In every case, GRAIN is consistently better than all baselines, indicating that key mismatch-mitigating design intent from the manual layout has been preserved in the migrated layout.

Look at placement and routing migration as a whole; the performance gap between GRAIN and the baseline widens at 28nm. As parasitics become more pronounced in advanced technology, the baseline's placement and routing lead to a performance collapse. In OTA at 28nm, [3] fix suffers severe gain loss (-42.13 dB), while [3]+Ours exhibits unstable PM (31.69°), both deviating from the original design goal despite higher bandwidth. In contrast, GRAIN achieves the closest overall match to the schematic at 28nm, with the smallest PM deviation ($+9.31°$) and leading CMRR/PSRR, demonstrating the benefit on performance by migrating design intent systematically. Notably, our placement migration algorithm achieves performance gains while simultaneously reducing both area and wire length. Compared to the baseline, the total area and total wire length are reduced by an average of 13.8% and 29.2%, respectively. This means that the layouts generated by our migration solution more closely resemble the compact and high-performance layouts achieved in manual layouts.

The migrated layouts of OTA are visually shown in Figure 5. Relative to the baseline [3], our placement reduces unused whitespace and preserves symmetry and align groups and cluster boundaries consistent with guard ring and shared N-well organization. The routed patterns show fewer unnecessary jogs and better regularity under our guide-based flow, aligning with the WL reductions and higher constraint satisfaction reported in Table I. At 28nm, the baseline exhibits broken symmetry and irregular detours around critical pairs, matching its degraded gain/PM, whereas our result maintains closer schematic performance. Overall, across nodes and designs, the proposed design-intent-driven layout migration framework yields LVS-clean results without manual fixes, preserves analog intent more faithfully, reduces layout area and wire length, and gets better post-layout metrics relative to the representative baseline.

V. Conclusion

This work introduced GRAIN, an analog layout migration framework featuring hierarchical placement and guide-based routing to preserve design intent. Evaluation on 65nm to 40nm/28nm migrations shows that GRAIN achieves 100% LVS-clean results without manual intervention, outperforming state-of-the-art methods by reducing area and wirelength by 13.8% and 29.2%, respectively. Despite scaling challenges in complex heuristics, GRAIN significantly advances automated EDA tools by providing a practical, stable solution for high-fidelity layout migration.

REFERENCES

[1] S. Hammouda *et al.*, "Chameleon art: A non-optimization based analog design migration framework," in *Proceedings of the 43rd annual Design Automation Conference*, 2006, pp. 885–888.

[2] S. H. Batterywala *et al.*, "Cell swapping based migration methodology for analog and custom layouts," in *9th International Symposium on Quality Electronic Design (isqed 2008)*. IEEE, 2008, pp. 450–455.

[3] H.-Y. Chi *et al.*, "A style-based analog layout migration technique with complete routing behavior preservation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, pp. 2556–2567, 2021.

[4] P.-C. Pan *et al.*, "A fast prototyping framework for analog layout migration with planar preservation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, pp. 1373–1386, 2015.

[5] H.-Y. Chi *et al.*, "Achieving routing integrity in analog layout migration via cartesian detection lines," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–6.

[6] M. P.-H. Lin *et al.*, "Achieving analog layout integrity through learning and migration," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–8.

[7] Y.-P. Weng *et al.*, "Fast analog layout prototyping for nanometer design migration," in *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2011, pp. 517–522.

[8] M. P.-H. Lin *et al.*, "Augmenting slicing trees for analog placement," in *2012 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, 2012, pp. 57–60.

[9] B. Xu *et al.*, "MAGICAL: Toward Fully Automated Analog IC Layout Leveraging Human and Machine Intelligence: Invited Paper," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. Westminster, CO, USA: IEEE, Nov. 2019, pp. 1–8.

[10] H. Chen *et al.*, "MAGICAL: An Open- Source Fully Automated Analog IC Layout System from Netlist to GDSII," *IEEE Design & Test*, vol. 38, pp. 19–26, 2021.

[11] H. Chen *et al.*, "MAGICAL 1.0: An Open-Source Fully-Automated AMS Layout Synthesis Framework Verified With a 40-nm 1GS/s $\Sigma\Delta$ ADC," in *2021 IEEE Custom Integrated Circuits Conference (CICC)*. Austin, TX, USA: IEEE, Apr. 2021, pp. 1–2.

[12] K. Kunal *et al.*, "ALIGN: Open-Source Analog Layout Automation from the Ground Up," in *Proceedings of the 56th Annual Design Automation Conference 2019*. Las Vegas NV USA: ACM, Jun. 2019, pp. 1–4.

[13] T. Dhar *et al.*, "ALIGN: A System for Automating Analog Layout," *IEEE Design & Test*, vol. 38, pp. 8–18, Apr. 2021.

[14] S. S. Sapatnekar, "The ALIGN Automated Analog Layout Engine: Progress, Learnings, and Open Issues," in *Proceedings of the 2023 International Symposium on Physical Design*. Virtual Event USA: ACM, Mar. 2023, pp. 101–102.

[15] X. Gao *et al.*, "Joint placement optimization for hierarchical analog/mixed-signal circuits," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2024, pp. 1–9.

[16] H. Zhang *et al.*, "Sageroute: Synergistic analog routing considering geometric and electrical constraints with manual design compatibility," in *DATE*, 2023, pp. 1–6.

[17] H. Zhang *et al.*, "Sageroute2.0: Hierarchical analog and mixed signal routing considering versatile routing scenarios," in *DATE*, 2024, pp. 1–6.

[18] J. Crossley *et al.*, "BAG: A designer-oriented integrated framework for the development of AMS circuit generators," in *ICCAD*, Nov. 2013, pp. 74–81, iSSN: 1558-2434.

[19] S. Youssef *et al.*, "A python-based layout-aware analog design methodology for nanometric technologies," in *2011 IEEE 6th International Design and Test Workshop (IDT)*. IEEE, 2011, pp. 62–67.

[20] E. Chang *et al.*, "Bag2: A process-portable framework for generator-based ams circuit design," in *2018 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2018, pp. 1–8.

[21] Z. Ye *et al.*, "Ted: A python-based analog design environment for agile circuit development," in *2023 International Symposium of Electronics Design Automation (ISEDA)*. IEEE, 2023, pp. 5–10.

[22] J. Han *et al.*, "Laygo: A template-and-grid-based layout generation engine for advanced cmos technologies," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, pp. 1012–1022, 2021.

[23] S. Bhattacharya *et al.*, "Hierarchical extraction and verification of symmetry constraints for analog layout automation," in *ASP-DAC 2004: Asia and South Pacific Design Automation Conference 2004 (IEEE Cat. No.04EX753)*, 2004, pp. 400–405.

[24] S. Hammouda *et al.*, "Chameleon art: a non-optimization based analog design migration framework," in *2006 43rd ACM/IEEE Design Automation Conference*, 2006, pp. 885–888.

[25] S. Bhattacharya *et al.*, "Multilevel symmetry-constraint generation for retargeting large analog layouts," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 945–960, 2006.

[26] S. Wang *et al.*, "Analog layout retargeting with geometric programming and constrains symbolization method," in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2012, pp. 353–356.

[27] L. Zhang *et al.*, "Parasitic-aware optimization and retargeting of analog layouts: A symbolic-template approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 791–802, 2008.

[28] H. Zhou and J. Wang, "Acg-adjacent constraint graph for general floorplans," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings.*, 2004, pp. 572–575.

[29] E. Young *et al.*, "Placement constraints in floorplan design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, pp. 735–745, 2004.

[30] M. Moffitt *et al.*, "Constraint-driven floorplan repair," in *2006 43rd ACM/IEEE Design Automation Conference*, 2006, pp. 1103–1108.

[31] C. Kodama *et al.*, "A novel encoding method into sequence-pair," in *2004 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 5, 2004, pp. V–V.

[32] A. Patyal *et al.*, "Analog placement with current flow and symmetry constraints using pcp-sp," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[33] Y.-P. Weng *et al.*, "Fast analog layout prototyping for nanometer design migration," in *2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2011, pp. 517–522.

[34] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.