CAN GROUND-CAPACITANCE PREDICTORS GENERALIZE ACROSS CIRCUIT DOMAINS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Producing circuit layouts requires substantial engineering effort, and failed post-layout verification can trigger further layout, extraction, and simulation iterations. These costs, together with restricted access to circuit intellectual property, make labeled parasitic data scarce when entering a new circuit regime. Learning-based prediction could provide earlier electrical feedback, but its usefulness in this cold-start setting depends on reusing data from other circuit domains. We study target-label-free, node-level ground-capacitance prediction across Digital, Analog, and SRAM circuits, where similar graph structures can carry different physical contexts and loading behavior. We propose XRC-Former, a graph Transformer that preserves circuit context through domain-sensitive encoding, reuses structural prototypes through shared memory, and aligns only a designated invariant branch while retaining a non-aligned residual for prediction. Under leave-one-domain-out evaluation, using source labels for training and source validation for model selection, XRC-Former achieves the best target-domain R^2 among 14 task-adapted baselines. It remains positive in the difficult Analog-target setting, where most baselines obtain negative R^2. Ablations, embedding diagnostics, and supervised within-domain references characterize the roles and scope of the proposed transfer mechanism. The results support cross-domain GC prediction as a step toward reusable parasitic screening before the final layout stage, rather than a replacement for signoff verification.

1 INTRODUCTION

Parasitic effects are a major source of mismatch between schematic-level intent and post-layout circuit behavior. Obtaining reliable parasitic labels requires more than running an extraction tool: engineers must first enter a physical design stage to produce a valid layout. Custom Analog layout involves substantial manual placement, routing, and matching effort for sensitive devices, while Digital back-end implementation requires engineering effort for physical constraint adjustments and engineering change orders (ECOs) to make the design converge. Only after design rule check (DRC), layout-versus-schematic (LVS) check, and timing analysis are passed, the resulting layout can then be extracted through post-layout simulation. If the design fails its specifications, engineers may need to revise the circuit or layout and repeat this cycle. Early parasitic feedback is therefore valuable because it could expose problematic loads before further layout and verification effort is committed. Learning-based predictors aim to provide such feedback from the circuit information available at an early (front-end) design stage (Shook et al., 2020; Ren et al., 2020; Yoon et al., 2025); this role complements rather than replaces final extraction and signoff.

These physical-design costs also make labeled circuit data expensive to obtain. Circuit intellectual property (IP) embodies substantial design investment, and access to a Digital RTL design does not by itself provide a verified layout or parasitic labels. Data collected by different companies and teams are further tied to their design families, process technologies, and implementation flows. A team entering a new circuit regime may therefore have only one labeled reference design, or none, even when it has accumulated data for other types of circuits. The practical question is whether those existing data remain useful: can a predictor reuse knowledge from available circuit domains before sufficient labels exist to train a target-specific model? Cross-domain prediction addresses this cold-start setting, rather than assuming that it should replace a well-trained specialist when representative target labels are already available.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Transfer is challenging because Digital logic, Analog blocks, and SRAM arrays differ in graph structure, physical context, and loading statistics. Digital designs often contain standard-cell and routing patterns, Analog circuits depend on sensitive local electrical and physical context, and SRAM arrays exhibit dense repeated structures. Prior learning-based parasitic methods mainly target specific circuit regimes or task settings. MLParest (Shook et al., 2020) and ParaGraph (Ren et al., 2020) focus on in-house analog circuits parasitics or device-parameter estimation, ParaFormer (Yoon et al., 2025) studies pre-routing RC prediction with heterogeneous graph modeling for digital circuits, DLPL-Cap (Shen et al., 2024) targets SRAM capacitance prediction, and CirGPS (Shen et al., 2025b) and CircuitGCL (Shen et al., 2025a) investigate data-efficient parasitic transfer for AMS-style circuits. Generic GNNs, graph Transformers, graph pre-training methods, and multi-domain graph models (Kipf & Welling, 2017; Veličković et al., 2018; Hamilton et al., 2017; Wu et al., 2023; Deng et al., 2024; Hu et al., 2020a; Qiu et al., 2020; Wang et al., 2025; Yuan et al., 2025) provide useful ingredients, but do not directly resolve how circuit-specific predictive context should be retained while transferring across these macro domains.

We study this question through node-level ground-capacitance (C_g) prediction on NET nodes of typed circuit component graphs. C_g provides a common, physically meaningful target because it contributes to electrical loading, delay, and switching energy across Digital, Analog, and SRAM circuits. Fully shared representations can suppress useful circuit context, whereas fully domain-specific models cannot readily reuse recurring local structures. The intended use is parasitic screening once the required graph and context features are available before the physical design is produced; the present study evaluates prediction and transfer accuracy rather than a measured reduction in design iterations.

Our contributions are threefold. (1) We formulate cross-macro-domain NET-node C_g prediction under a target-label-free leave-one-domain-out protocol, addressing the cold start of a new circuit regime. (2) We propose **XRC-Former**, a graph Transformer that couples circuit-derived encoding, shared structural memory, and selective invariant-space alignment to preserve circuit context, reuse recurring patterns, and align only a designated transferable branch while retaining a non-aligned residual for prediction. (3) We evaluate transfer across Digital, Analog, and SRAM against 14 task-adapted baselines, complemented by supervised within-domain references, ablations, embedding diagnostics, training dynamics, and few-shot adaptation.

We claim that the proposed method can be used not only for C_g prediction but also for coupling capacitance and resistance prediction that we leave as future work.

2 Related Work

2.1 Learning-Based and Transferable Parasitic Prediction

Parasitic resistance and capacitance arise from the physical realization of devices and interconnects. Finite conductor and contact resistance introduce voltage drops and, together with capacitance, affect delay and settling. Ground capacitance (C_g) represents loading to the reference node, whereas coupling capacitance (C_c) links distinct signal nets and can introduce crosstalk and activity-dependent loading. These effects influence timing, switching energy, and Analog or SRAM behavior; no single parameter alone determines post-layout performance. Predicting them earlier could help prioritize design changes before another costly layout–extraction–verification cycle.

Learning-based parasitic prediction estimates layout-dependent electrical quantities before full extraction, enabling earlier feedback during circuit design. For custom circuits, MLParest (Shook et al., 2020) predicts effective resistance and capacitance from schematic-level net features, while ParaGraph (Ren et al., 2020) converts schematics into graphs and predicts layout parasitics and device parameters with GNNs. For pre-routing or memory-style settings, ParaFormer (Yoon et al., 2025) combines a heterogeneous GNN and graph Transformer for RC prediction, and DLPL-Cap (Shen et al., 2024) targets pre-layout capacitance prediction for SRAM designs. For transferable AMS parasitic estimation, CircuitGPS (Shen et al., 2025b) uses heterogeneous graphs and few-shot adaptation, while CircuitGCL (Shen et al., 2025a) combines graph contrastive learning with label rebalancing. These methods motivate data-driven and data-efficient parasitic estimation, but they are mainly designed for specific circuit regimes rather than cross-domain node-level ground-capacitance prediction across Digital, Analog, and SRAM circuits.

We focus on C_g as a common NET-node target across the three macro domains, not as a substitute for complete RC characterization. The transferable representation and memory mechanisms provide a basis for extending the framework to other parasitics. Such extensions require task-specific outputs: C_c is naturally a net-pair prediction problem, while resistance requires edge- or segment-level supervision. They also require appropriate geometry features and physical constraints. This paper validates C_g transfer; however, the propsoed method can be adapted for resistance and coupling-capacitance prediction, which remains future extensions.

2.2 Graph Backbones for Circuit Representation Learning

Because circuit structures are naturally represented as graphs, we compare against representative graph backbones adapted to the same circuit prediction task (Sheehan, 1999). GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), GraphSAGE (Hamilton et al., 2017), and PNA (Corso et al., 2020) cover standard message-passing designs, from graph convolution and attention to inductive aggregation and multi-aggregator propagation. SGFormer (Wu et al., 2023) and PolyNormer (Deng et al., 2024) provide graph Transformer-style baselines. These models test whether generic graph aggregation or attention mechanisms are sufficient for cross-domain transfer, but they do not explicitly separate reusable circuit structure from domain-conditioned circuit context.

2.3 Multi-Domain Graph Transfer Baselines

Multi-domain graph learning studies how graph knowledge can transfer across heterogeneous domains. In our experiments, the most relevant baselines are BRIDGE (Yuan et al., 2025), which studies bounded multi-domain graph foundation modeling with domain-invariant alignment and selective knowledge transfer, and MDGFM (Wang et al., 2025), which improves multi-domain graph transfer through topology alignment and graph refinement. These methods target cross-domain graph generalization and therefore provide natural comparisons for leave-one-domain-out circuit transfer. However, their original settings are general graph-learning benchmarks rather than parasitics regression over typed circuit graphs with Digital, Analog, and SRAM-specific physical distributions.

3 Problem Formulation

We represent each circuit component as a heterogeneous graph $G = (V, E, \mathbf{X}, \mathbf{t}, \mathbf{e}, D)$, where V is the node set, E is the edge set, $\mathbf{X} \in \mathbb{R}^{|V| \times d_{\text{in}}}$ is the node-feature matrix, $\mathbf{t}$ denotes node types, $\mathbf{e}$ denotes observed edge types, and $D \in \mathcal{D}$ is the macro-domain label. In this work, $\mathcal{D} = \{\text{Digital}, \text{Analog}, \text{SRAM}\}$. We formulate the problem at the component-graph level because cross-domain transfer must preserve both typed circuit connectivity and macro-domain identity.

Let $\mathcal{M}_{\text{net}} = \{v \in V : \mathbf{t}_v = \text{NET}\}$ denote the set of NET nodes with ground-capacitance labels. The node-level GC prediction task is defined in Equation 1:

$$\hat{c}_v = f_\theta(G, v), \qquad v \in \mathcal{M}_{\text{net}}. \tag{1}$$

Here c_v is the normalized ground-capacitance label and $\hat{c}_v$ is the prediction. Training optimizes the GC regression loss in Equation 2:

$$\mathcal{L}_{GC} = \frac{1}{|\mathcal{M}_{\text{net}}|} \sum_{v \in \mathcal{M}_{\text{net}}} \rho(\hat{c}_v - c_v). \tag{2}$$

The learning problem is cross-domain generalization over macro domains. We train on source domains $\mathcal{D}_S \subset \mathcal{D}$ and evaluate on held-out target domains $\mathcal{D}_T = \mathcal{D} \setminus \mathcal{D}_S$. We report two protocols. **Leave-one-domain-out (LODO) transfer** trains on two domains and evaluates on the held-out third domain without training data. **Few-shot adaptation** starts from the same split and adds a small number of labeled target-domain support components for adaptation. Appendix A summarizes the notation used throughout the paper.

4 Method

XRC-Former targets cross-domain node-level C_g prediction on heterogeneous circuit graphs. Its design follows three principles: preserve domain-conditioned circuit context, reuse recurring struc-

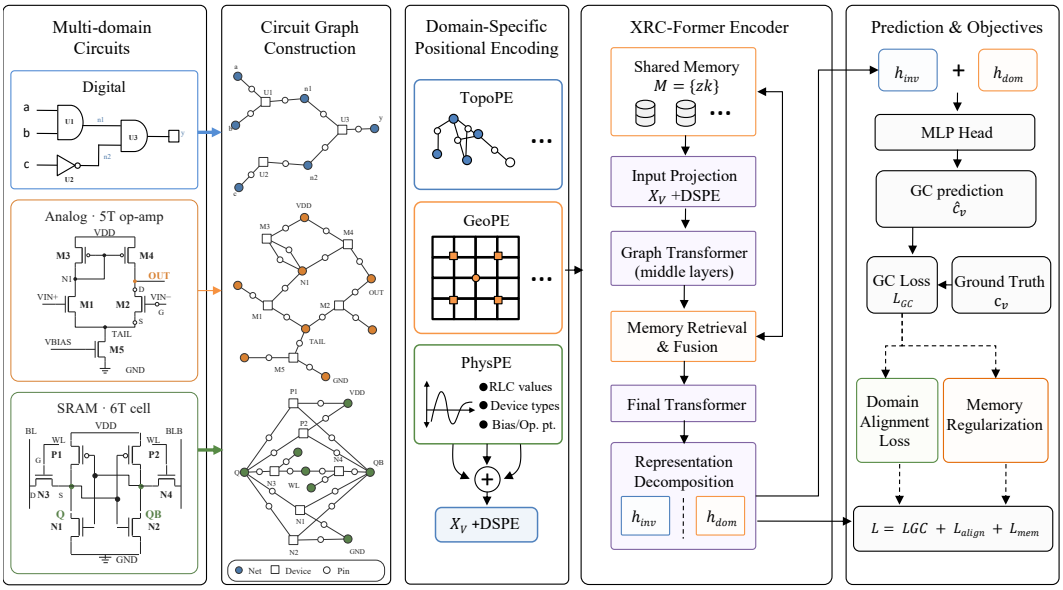


Figure 1: Overview of XRC-Former. The model preserves domain-sensitive circuit context, retrieves shared structural prototypes, and aligns the invariant subspace for cross-domain NET-node GC regression.

tural patterns, and align only the transferable part of the representation. Figure 1 summarizes the architecture.

Given a circuit component graph $G = (V, E, \mathbf{X}, \mathbf{t}, \mathbf{e}, D)$ with domain label $D \in \mathcal{D}$, XRC-Former first injects domain-sensitive circuit encodings and processes the graph with masked Transformer layers. The resulting node states query a shared memory bank of reusable structural prototypes. A final Transformer layer integrates the retrieved memory context, after which the representation is decomposed into invariant and domain-conditioned components for C_g prediction on NET nodes.

4.1 Preserve: Domain-Sensitive Circuit Encoding

A single shared positional encoding can erase the fact that the same graph motif may carry different physical meaning in Digital, Analog, and SRAM circuits. XRC-Former therefore injects domain-sensitive circuit information before message passing so that subsequent attention operates on node states that already reflect macro-domain context.

For each node v, the encoding combines topological, geometric, and physics-aware descriptors:

$$\mathbf{p}_v^{\mathrm{dspe}} = \alpha_t \phi^{\mathrm{topo}}(\mathbf{p}_v^{\mathrm{topo}}) + \alpha_g \phi_D^{\mathrm{geo}}(\mathbf{p}_v^{\mathrm{geo}}) + \alpha_p \phi_D^{\mathrm{phys}}(\mathbf{p}_v^{\mathrm{phys}}). \tag{3}$$

Here $\mathbf{p}_v^{\mathrm{topo}}$, $\mathbf{p}_v^{\mathrm{geo}}$, and $\mathbf{p}_v^{\mathrm{phys}}$ are node-level descriptor vectors; the mappings ϕ project them into a common d-dimensional hidden space. The subscript D denotes domain-conditioned mappings, and $\alpha_t, \alpha_g, \alpha_p$ scale the three contributions. Their sum is added to the input projection.

Topological positional encoding (TopoPE) captures graph position and local topology and is shared across domains, since notions such as connectivity, random-walk structure, and coarse graph position remain meaningful in every macro domain. Geometry positional encoding (GeoPE) captures routing- or layout-derived geometry when available, but it is domain-specific because geometry availability and reliability differ across Digital, Analog, and SRAM. Physics positional encoding (PhysPE) captures local electrical context through proxy features and edge-type statistics. These physics descriptors are auxiliary context features rather than prediction targets: they help the model distinguish physically different neighborhoods before regression, but the supervision still remains C_g on NET nodes.

XRC-Former also injects this information into attention through a compact structural bias,

$$\mathbf{B}^{\mathrm{base}} = \beta_{\mathrm{adj}} \mathbf{A} + \beta_{\mathrm{pos}} \bar{\mathbf{P}}^{\mathrm{dspe}} (\bar{\mathbf{P}}^{\mathrm{dspe}})^{\top}, \tag{4}$$

where $\mathbf{A}$ is the adjacency matrix and $\bar{\mathbf{P}}^{\mathrm{dspe}}$ denotes row-normalized DSPE vectors. The scalar coefficients β_{adj} and β_{pos} control the adjacency and DSPE-similarity biases, respectively. These biases reweight attention; the structural mask determines which node pairs may interact. In this way, the preserve module keeps domain-conditioned circuit meaning explicit before attention begins to mix information across the graph.

4.2 Reuse: Shared Structural Memory

Cross-domain transfer does not require every hidden state to collapse into one shared domain-invariant representation. Still, circuit graphs from different macro domains often contain recurring local contexts, such as repeated device–net neighborhoods, fanout motifs, or array-like structures. XRC-Former captures these reusable contexts with a shared memory bank $\mathbf{Z} = \{\mathbf{z}_k\}_{k=1}^{K}$, where each slot acts as a latent structural prototype learned across source-domain graphs.

Let $\mathbf{H}^{\mathrm{base}}$ denote the hidden states before the final Transformer layer. Each node softly retrieves memory slots by

$$\boldsymbol{\beta} = \mathrm{softmax}(\mathbf{H}^{\mathrm{base}}\mathbf{Z}^{\top}/\tau), \qquad \mathbf{M}^{\mathrm{mem}} = \boldsymbol{\beta}\mathbf{Z}, \qquad \mathbf{H}^{\mathrm{fused}} = \mathbf{H}^{\mathrm{base}} + \mathbf{M}^{\mathrm{mem}}. \tag{5}$$

Here $\tau > 0$ is the temperature, and softmax operates over the K memory slots for each node; a smaller temperature yields sharper assignments. For n nodes, $\mathbf{H}^{\mathrm{base}} \in \mathbb{R}^{n\times d}$ and the stacked prototype matrix $\mathbf{Z} \in \mathbb{R}^{K\times d}$ give assignments $\boldsymbol{\beta} \in \mathbb{R}^{n\times K}$. The assignments measure how strongly each node matches each prototype, the retrieved memory matrix $\mathbf{M}^{\mathrm{mem}}$ aggregates those prototypes, and the residual fusion keeps memory as additional reusable context rather than a replacement for node features.

Memory also shapes the final attention pattern. Nodes that retrieve similar prototypes should be allowed to interact more strongly, even when their raw local neighborhoods differ across domains. XRC-Former therefore adds a memory-induced attention bias,

$$\mathbf{B}^{\mathrm{mem}} = \gamma\, \boldsymbol{\beta}\boldsymbol{\beta}^{\top}, \tag{6}$$

where the scalar γ scales the memory-induced bias and $(\boldsymbol{\beta}\boldsymbol{\beta}^{\top})_{uv}$ measures agreement between the memory assignments of nodes u and v. This bias turns the memory bank into an interface for shared structural reasoning rather than a passive cache of extra parameters.

Memory slots are updated by hard-assignment exponential moving average (Appendix C) to track recurring source-domain hidden patterns.

4.3 Align: Selective Representation Alignment and Prediction

XRC-Former aligns only the reusable subspace, since forcing the full hidden state to be domain-invariant can remove GC-relevant domain context.

Final fused node representation. The final Transformer layer fuses retrieved prototypes with node states and combines DSPE and memory biases:

$$\mathbf{H}^{(L)} = \mathrm{GT}_L\big(\mathbf{H}^{\mathrm{base}} + \mathbf{M}^{\mathrm{mem}}; \mathbf{B}^{\mathrm{base}} + \mathbf{B}^{\mathrm{mem}}, \mathcal{A}_{\mathrm{mask}}\big)\,. \tag{7}$$

Here GT_L is the last of L Transformer layers, and $\mathcal{A}_{\mathrm{mask}}$ is the structural attention mask specifying permitted node pairs; masked-out pairs receive no attention weight.

Representation decomposition. After the final Transformer layer, each node state is decomposed into invariant and domain-conditioned components:

$$\mathbf{h}_v^{\mathrm{inv}} = \mathbf{W}_{\mathrm{inv}}\mathbf{h}_v^{(L)}, \qquad \mathbf{h}_v^{\mathrm{dom}} = \mathbf{W}_{\mathrm{dom}}\mathbf{h}_v^{(L)}, \qquad \hat{c}_v = \mathrm{MLP}(\mathbf{h}_v^{\mathrm{inv}} + \mathbf{h}_v^{\mathrm{dom}}). \tag{8}$$

The superscripts inv and dom denote the invariant and domain-conditioned branches. The learnable linear projections $\mathbf{W}_{\mathrm{inv}}$ and $\mathbf{W}_{\mathrm{dom}}$ produce compatible hidden vectors; only the invariant branch receives MMD alignment, while both contribute to prediction.

Selective invariant-space alignment. Alignment is applied only to source-domain invariant embeddings. Let $\mathbf{H}_i^{\text{inv}}$ denote a subsampled set of NET-node invariant embeddings from source domain i, and let $\mathcal{P}$ be the set of source-domain pairs in the current training fold. XRC-Former minimizes

$$\mathcal{L}_{align} = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \text{MMD}^2(\mathbf{H}_i^{\text{inv}}, \mathbf{H}_j^{\text{inv}}). \tag{9}$$

This loss makes the reusable subspace less domain-separable without forcing domain-specific residuals to disappear. The selectivity matters: the model is encouraged to align what should transfer, not everything that might still carry GC-relevant domain context.

Prediction and objective. To keep memory retrieval consistent with the learned structural prototypes, we regularize the pre-final hidden state against its retrieved memory context,

$$\mathcal{L}_{mem} = \frac{1}{|\mathcal{M}_{\text{net}}|} \sum_{v\in\mathcal{M}_{\text{net}}} \|\mathbf{h}_v^{\text{base}} - \mathbf{m}_v\|_2^2, \tag{10}$$

where $\mathbf{m}_v$ is the retrieved memory vector for node v. This term encourages the hidden space and the prototype space to remain compatible.

The final training objective combines C_g regression, selective alignment, and memory regularization:

$$\mathcal{L} = \mathcal{L}_C + \lambda_{align}\mathcal{L}_{align} + \lambda_{mem}\mathcal{L}_{mem}. \tag{11}$$

Here $\mathcal{L}_C$ is the regression loss on NET nodes' capacitance, while $\mathcal{L}_{align}$ and $\mathcal{L}_{mem}$ control the reusable subspace and the prototype interface, respectively. Detailed optimization settings, hyperparameters, and optional variants are deferred to Appendix F; the key main-text point is that XRC-Former predicts C_g from both reusable and domain-conditioned information, while aligning only the part that should transfer across macro domains.

5 Experiments

We distinguish three supervision regimes. Supervised within-domain training provides a reference when target labels are available (Appendix E.1); few-shot adaptation examines transfer with limited labeled target support (Appendix E.2); and source-only leave-one-domain-out (LODO) evaluation tests the cold start without target-domain supervision. These settings have different label budgets and should not be conflated. The main text focuses on LODO through four research questions:

- **RQ1:** Does XRC-Former improve held-out macro-domain GC prediction?
- **RQ2:** Does the invariant subspace become more comparable across domains?
- **RQ3:** Which components are necessary for transfer?
- **RQ4:** Does better source-domain optimization translate into better target-domain transfer?

5.1 Experimental Settings

Task and domains. The task is node-level C_g regression on heterogeneous circuit component graphs, as defined in Section 3. Our circuit data are sourced from ParasGB (Zou et al., 2026). We consider three macro domains—Digital, Analog, and SRAM—and evaluate transfer across all three held-out settings: Analog+SRAM→Digital, Digital+Analog→SRAM, and Digital+SRAM→Analog.

Baselines. We compare against four categories of baselines. Generic GNN backbones include GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), GraphSAGE (Hamilton et al., 2017), and PNA (Corso et al., 2020). Graph Transformer backbones include SGFormer (Wu et al., 2023) and PolyNormer (Deng et al., 2024). Multi-domain graph models include MDGFM (Wang et al., 2025) and BRIDGE (Yuan et al., 2025). Circuit/parasitic predictors include MLParest, ParaGraph, ParaFormer, DLPL-Cap, CirGPS, and CircuitGCL. Together, these comparisons test whether the proposed gains come from using a stronger generic graph encoder, a generic multi-domain transfer model, or a circuit-oriented parasitic predictor.

Table 1: Leave-one-domain-out cross-domain ground capacitance prediction measured by target-domain R^2 across Digital, Analog, and SRAM macro domains. Models are trained on two source domains and evaluated on the held-out target domain without target-domain training labels. Higher is better; negative R^2 indicates worse performance than predicting the target-domain mean. Best is shown in bold and runner-up is underlined.

Category	Method	A+S→D	D+A→S	D+S→A
GNN	GCN (Kipf & Welling, 2017)	0.6926 ± 0.2567	0.0808 ± 0.0993	-0.5858 ± 0.0698
	GAT (Veličković et al., 2018)	0.7034 ± 0.1390	0.2764 ± 0.0743	$\underline{-0.1783 \pm 0.1121}$
	GraphSAGE (Hamilton et al., 2017)	0.7032 ± 0.0781	0.2940 ± 0.0337	-0.2288 ± 0.1226
	PNA (Corso et al., 2020)	0.7416 ± 0.1203	0.1952 ± 0.0781	-0.3034 ± 0.1321
Graph Transformer	SGFormer (Wu et al., 2023)	$\underline{0.8032 \pm 0.0608}$	0.1778 ± 0.0663	-0.3281 ± 0.2700
	PolyNormer (Deng et al., 2024)	0.7442 ± 0.0462	0.3194 ± 0.0447	-0.1934 ± 0.0600
Multi-Domain GFM	MDGFM (Wang et al., 2025)	0.5899 ± 0.1389	0.1080 ± 0.0508	-0.5757 ± 0.1316
	BRIDGE (Yuan et al., 2025)	0.7092 ± 0.3019	0.1419 ± 0.1668	-0.5699 ± 0.6115
Parasitic GNN	ParaFormer (Yoon et al., 2025)	0.7126 ± 0.0663	0.2616 ± 0.0389	-0.8502 ± 0.1442
	MLParest (Shook et al., 2020)	0.5441 ± 0.0878	-0.0316 ± 0.0235	-0.7587 ± 0.1580
	DLPL-Cap (Shen et al., 2024)	0.4354 ± 0.0878	0.0123 ± 0.0235	-0.3649 ± 0.1580
	ParaGraph (Ren et al., 2020)	0.7373 ± 0.0257	$\underline{0.3548 \pm 0.0660}$	-0.4198 ± 0.1485
	CircuitGPS (Shen et al., 2025b)	0.7613 ± 0.0299	0.2988 ± 0.0263	-0.6189 ± 0.1408
	CircuitGCL (Shen et al., 2025a)	0.7777 ± 0.0433	0.2358 ± 0.0312	-0.2794 ± 0.4152
Ours	XRC-Former (ours)	$\mathbf{0.8171 \pm 0.0848}$	$\mathbf{0.5567 \pm 0.1002}$	$\mathbf{0.4245 \pm 0.1991}$

Repeated runs and appendix details. All reported values are means $\pm$ standard deviations over repeated runs. Because the archived manuscript results are aggregate repeated-run summaries rather than paired per-run tables for every method, we treat these deviations as variability summaries rather than formal significance tests. Appendix D provides the full split, metric, repeated-run, and baseline-adaptation details, and Appendix D.1 gives the benchmark construction details.

5.2 Research Question 1 (RQ1): Held-Out Macro-Domain GC Prediction

RQ1 asks whether the learned predictor remains useful when deployed on an unseen circuit regime rather than the regimes used for model development. Table 1 therefore serves as the main result: target-domain R^2 when each model is trained on two macro domains and evaluated on the held-out third. We have the following observations: ① **XRC-Former is consistently strongest across targets.** It achieves the best target-domain R^2 on all three folds, so the gain is not limited to one favorable held-out domain. ② **Analog is the clearest stress test.** Under Digital+SRAM→Analog, most baselines obtain negative R^2, meaning that they perform worse than predicting the Analog-domain mean. XRC-Former remains positive, which is the strongest evidence that the model handles severe macro-domain mismatch. ③ **The gap is not explained by one baseline family.** Generic graph backbones are competitive on easier targets, multi-domain graph models provide general transfer machinery, and circuit-specific parasitic predictors encode relevant inductive biases, but none transfers reliably across all three domains. This supports the need to separate reusable structure from domain-conditioned context. Extended MAE and RMSE results are reported in Appendix E.5 and provide a complementary absolute-error view.

5.3 Research Question 2 (RQ2): Does Alignment Improve Representation Comparability?

RQ2 is a mechanism check rather than a second main result. It asks whether the invariant subspace learned by XRC-Former becomes less separable by macro domain, which would make representations more comparable across Digital, Analog, and SRAM. We have the following observations: ① **Figure 2 shows that domain separability decreases.** The domain silhouette drops from 0.276 to 0.159, meaning that the invariant embeddings become less separable by Digital, Analog, and SRAM domain. ② **This is mechanism evidence rather than standalone performance evidence.** The silhouette uses domain labels rather than C_g classes, and a lower value is better because the invariant subspace should be less domain-separable. It should therefore be read together with Table 1 and the ablations, because prediction still uses domain-specific residuals even as the invariant subspace becomes more comparable across domains. Few-shot target adaptation results for all three held-out

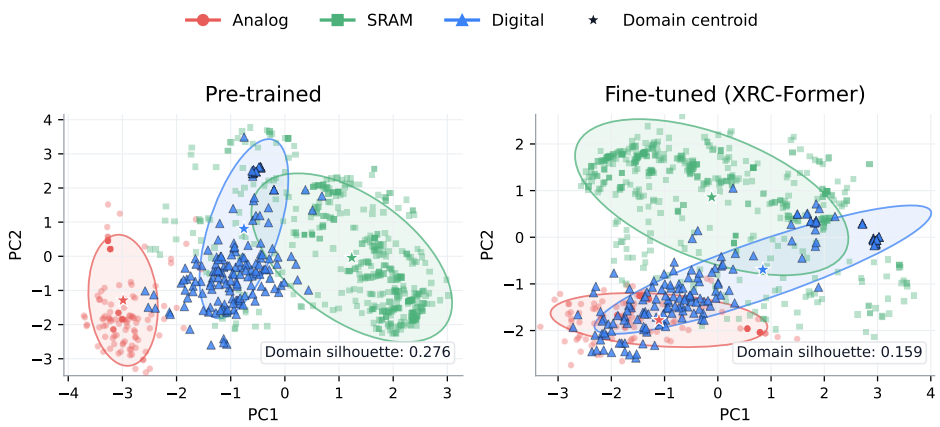


Figure 2: Invariant embedding alignment before and after XRC-Former training. Embeddings are projected onto the first two principal components, and colors indicate circuit domains rather than GC labels. We compute domain silhouette using Analog/SRAM/Digital as labels; lower values indicate that domain-invariant embeddings are less separable by domain and therefore better aligned. XRC-Former reduces the domain silhouette from 0.276 to 0.159.

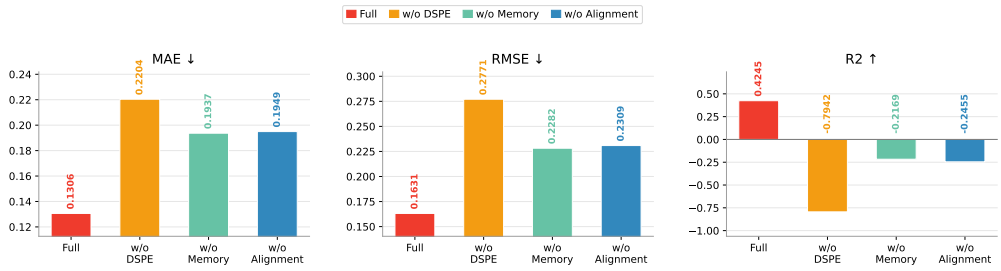


Figure 3: Ablation study for cross-domain ground capacitance prediction on the Analog target domain. Models are trained on Digital+SRAM source domains and evaluated on labeled Analog NET nodes. Removing DSPE, memory, or alignment substantially degrades MAE, RMSE, and R^2, showing that all three components are important for transferable C_g regression under severe domain shift.

domains are reported in Appendix E.2. We keep them in the appendix because the main text focuses on transfer to an unseen macro domain without target-domain training labels, but they are consistent with the same picture: limited target supervision improves all methods, while XRC-Former starts from a stronger transfer point and remains competitive after adaptation.

5.4 Research Question 3 (RQ3): Which Components Are Necessary?

RQ3 asks which transfer components matter once the held-out domain becomes substantially different from the training domains. We run the main ablation on the Analog target because Digital+SRAM→Analog is the most demanding transfer setting in Table 1, making it the clearest place to test whether each component contributes meaningfully under severe macro-domain mismatch. We have the following observations in Figure 3: ① **Removing DSPE causes the largest drop.** This supports the need to preserve domain-conditioned structural and physical cues, especially on the Analog target where local context matters most. The result also clarifies why generic transfer machinery alone is insufficient: without domain-sensitive encoding, the model loses the circuit context needed to interpret shared graph motifs correctly. ② **Removing memory hurts transfer as well.** The drop shows that recurring structural prototypes add information beyond local encoding alone. In other words, preserving domain-conditioned cues is not enough; the model also benefits from an explicit interface for reusing cross-domain structural regularities. ③ **Removing alignment also degrades performance.** Making the reusable subspace more comparable across source domains remains important even when DSPE and memory are present. This complements Figure 2: transfer improves

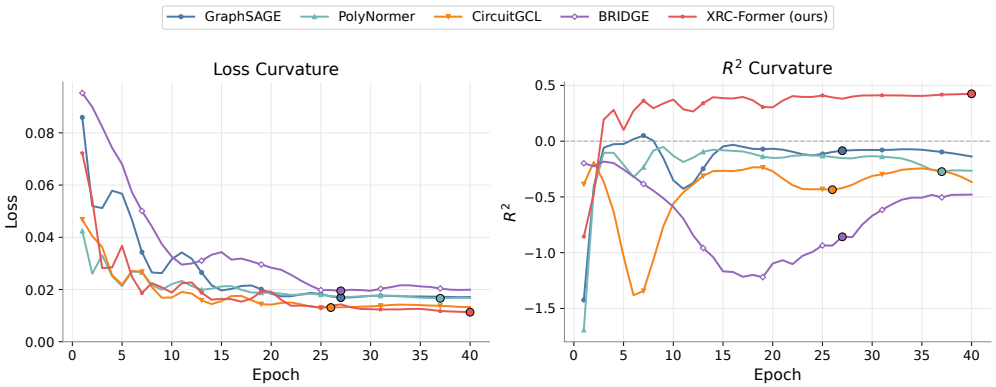


Figure 4: Training dynamics under source-only LODO transfer with Analog as the held-out target domain. Models are trained on Digital+SRAM source domains. The left panel reports source-validation MSE across epochs, while the right panel reports target-domain R^2 on labeled Analog NET nodes. Higher target-domain R^2 indicates better cross-domain GC prediction; negative R^2 indicates worse performance than predicting the target-domain mean.

most when the model both preserves domain-conditioned information and regularizes the part of the representation that should generalize. The full model's advantage comes from the combination rather than any single component, and Appendix E.4 reports additional target-domain ablations.

5.5 Research Question 4 (RQ4): Does Source Fit Predict Target Transfer?

RQ4 tests a different hypothesis: if low source-domain validation error were already sufficient for transfer, then methods that optimize source validation mean squared error (MSE) well should also achieve strong target-domain R^2. We test this on the Analog target because it most clearly reveals the difference between fitting available domains and transferring to a mismatched unseen domain. We have the following observations: ① **Source fit alone does not explain transfer.** Several baselines reduce source-validation MSE yet remain unstable or negative in target-domain R^2, so optimizing the available domains is not sufficient. The figure therefore rules out the simplest alternative explanation that XRC-Former wins merely because it fits the source domains more aggressively. ② **XRC-Former converts source-domain learning into target-domain utility more effectively.** It reaches positive target-domain R^2 earlier and maintains it more consistently over training, which matches the intended representation split between shared structure and domain-conditioned information. This is the key causal reading of Figure 4: transfer quality depends on how source learning is organized, not only on how low the source validation loss becomes. Additional Digital- and SRAM-target training dynamics are reported in Appendix E.3.

6 Conclusion

This paper studies cross-domain node-level ground-capacitance prediction as a controlled step toward reusable learning-based parasitic estimation across circuit regimes. The central challenge is not only to build a stronger graph predictor, but to decide which circuit information should transfer across macro domains and which information should remain domain-conditioned. XRC-Former addresses this challenge by combining domain-sensitive circuit encoding, shared structural memory, and selective invariant-space alignment within a graph Transformer.

Across Digital, Analog, and SRAM transfer settings, XRC-Former achieves the best target-domain R^2 among the compared baselines. Ablations, embedding diagnostics, and few-shot adaptation support the roles of the proposed components. More broadly, these results suggest that reusable parasitic prediction is not only an accuracy problem but a representation-transfer problem, requiring the model to separate reusable circuit structure from domain-conditioned physical context. Detailed scope boundaries and deployment caveats are discussed in Appendix G.

AI USE STATEMENT

Generative AI tools were used to assist with English-language editing and translation, LaTeX formatting, and manuscript revision planning. Additional assistance included organizing descriptions of existing experimental results and checking consistency among the manuscript, figures, tables, and evaluation-protocol descriptions. AI-generated suggestions served as drafting and editorial support rather than independent evidence for scientific claims. The authors retain full responsibility for the accuracy, originality, and integrity of the manuscript, including its methodology, experimental results, interpretations, and references.

REFERENCES

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Yuxuan Cao, Jiarong Xu, Carl Yang, Jiaan Wang, Yunchao Zhang, Chunping Wang, Lei Chen, and Yang Yang. When to pre-train graph neural networks? from data generation perspective! In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 142–153, 2023.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems*, volume 33, pp. 13260–13271, 2020.

Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In *International Conference on Learning Representations (ICLR)*, 2024.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. GPT-GNN: Generative pre-training of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1857–1867, 2020b.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. ZeroG: Investigating cross-dataset zero-shot transferability in graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1725–1735, 2024.

Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip S. Yu. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(6): 5879–5900, 2023a.

Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference*, pp. 417–428, 2023b.

Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. Learning to pre-train graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4276–4284, 2021.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1150–1160, 2020.

Haoxing Ren, George F Kokai, Walker J Turner, and Ting-Sheng Ku. Paragraph: Layout parasitics and device parameter prediction using graph neural networks. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2020.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Bernard N. Sheehan. Ticer: Realizable reduction of extracted rc circuits. In *Proceedings of the 1999 IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 1999.

Shan Shen, Dingcheng Yang, Yuyang Xie, Chunyan Pei, Bei Yu, and Wenjian Yu. Deep-learning-based pre-layout parasitic capacitance prediction on sram designs. In *Proceedings of the Great Lakes Symposium on VLSI 2024*, pp. 440–445, 2024.

Shan Shen, Shenglu Hua, Jiajun Zou, Jiawei Liu, Jianwang Zhai, Chuan Shi, and Wenjian Yu. Transferable parasitic estimation via graph contrastive learning and label rebalancing in ams circuits. In *2025 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9. IEEE, 2025a.

Shan Shen, Yibin Zhang, Hector Rodriguez Rodriguez, and Wenjian Yu. Few-shot learning on ams circuits and its application to parasitic capacitance prediction. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pp. 1–7. IEEE, 2025b.

Brett Shook, Prateek Bhansali, Chandramouli Kashyap, Chirayu Amin, and Siddhartha Joshi. Ml-parest: Machine learning based parasitic estimation for custom circuit design. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2020.

Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*, 2020.

Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. GPPT: Graph pre-training and prompt tuning to generalize graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1717–1727, 2022.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Shuo Wang, Bokui Wang, Zhixiang Shen, Boyan Deng, and Zhao Kang. Multi-domain graph foundation models: Robust knowledge transfer via topology alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 64806–64821, 2025.

Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. In *International Conference on Learning Representations*, 2022.

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Sgformer: Simplifying and empowering transformers for large-graph representations. *Advances in Neural Information Processing Systems*, 36:64753–64773, 2023.

Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2412–2429, 2023.

Jiarong Xu, Renhong Huang, Xin Jiang, Yuxuan Cao, Carl Yang, Chunping Wang, and Yang Yang. Better with less: A data-active perspective on pre-training graph neural networks. In *Advances in Neural Information Processing Systems*, volume 36, pp. 56946–56978, 2023.

Jongho Yoon, Jakang Lee, Donggyu Kim, Junseok Hur, and Seokhyeong Kang. Paraformer: A hybrid graph neural network and transformer approach for pre-routing parasitic rc prediction. In *Proceedings of the 30th Asia and South Pacific Design Automation Conference (ASPDAC '25)*, pp. 513–519. ACM, 2025. doi: 10.1145/3658617.3697599.

Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. Graph domain adaptation via theory-grounded spectral regularization. In *International Conference on Learning Representations*, 2023.

Haonan Yuan, Qingyun Sun, Junhua Shi, Xingcheng Fu, Bryan Hooi, Jianxin Li, and Philip S. Yu. How much can transfer? bridge: Bounded multi-domain graph foundation model with generalization guarantees. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 73604–73644, 2025.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-SAINT: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

Jiying Zhang, Xi Xiao, Long-Kai Huang, Yu Rong, and Yatao Bian. Fine-tuning graph neural networks via graph topology induced optimal transport. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 3730–3736, 2022.

Haihong Zhao, Aochuan Chen, Xiangguo Sun, Hong Cheng, and Jia Li. All in one and one for all: A simple yet effective method towards cross-domain graph pretraining. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4443–4454, 2024.

Qi Zhu, Carl Yang, Yidan Xu, Haonan Wang, Chao Zhang, and Jiawei Han. Transfer learning of graph neural networks with ego-graph information maximization. In *Advances in Neural Information Processing Systems*, volume 34, pp. 1766–1779, 2021.

Jiajun Zou, Jiawei Liu, Ao Liu, Junnong Tian, Yibin Zhang, Chengjie Liu, Yuxi Wang, Shan Shen, Wenhua Gu, Jun Yang, and Wenjian Yu. ParasGB: A graph benchmark suite for parasitic estimation on AMS circuits. *arXiv preprint arXiv:2607.23225*, 2026. doi: 10.48550/arXiv.2607.23225. URL https://arxiv.org/abs/2607.23225.

A NOTATION

This appendix collects the notation used throughout the main paper and supplement so that the objective definitions, algorithms, and experimental protocols can be read without repeatedly reintroducing symbols. We use D for a macro-domain label and $\mathcal{D} = \{\text{Digital}, \text{Analog}, \text{SRAM}\}$ for the set of macro domains. For edge information, $\mathbf{A}$ denotes edge existence, $\mathbf{e}_{uv}$ denotes the observed type on an existing edge $(u, v) \in E$, and $\mathbf{E}^{\text{type}}$ denotes the corresponding dense edge-type matrix with zeros on non-edges.

Table 2: Summary of key notation used throughout the paper and appendix.

Symbol	Definition
C_g	Ground parasitic capacitance
$G = (V, E)$	Circuit component graph
$\mathbf{X} \in \mathbb{R}^{n \times d_{\text{in}}}$	Node feature matrix
$\mathbf{A} \in \{0,1\}^{n \times n}$	Dense adjacency matrix (0=no edge, 1=edge; symmetric, with self-loops in attention)
$\mathbf{E}^{\text{type}} \in \{0,1,2\}^{n \times n}$	Dense edge-type matrix (0=no edge, 1=resistance, 2=capacitance)
$\mathbf{t}_v \in \{0,1,2\}$	Node type (0=NET, 1=DEVICE, 2=PIN)
$\mathbf{e}_{uv} \in \{0,1\}$	Observed edge type on $(u, v) \in E$ (0=resistance, 1=capacitance)
$\mathcal{M}_{\text{net}}$	Set of NET nodes with GC supervision
$c_v \in \mathbb{R}$	Normalized ground-capacitance label at node v
$\hat{c}_v \in \mathbb{R}$	Predicted ground capacitance at node v
$D \in \mathcal{D}$	Macro-domain label
$\mathcal{D} = \{\text{Digital, Analog, SRAM}\}$	Set of macro circuit domains
$\mathcal{D}_S$	Source macro domains used for training
$\mathcal{D}_T$	Held-out target macro domain(s)
d	Hidden dimension
L	Number of Transformer layers
h	Number of attention heads
K	Number of memory slots
$\mathbf{Z} = \{\mathbf{z}_k\}_{k=1}^{K} \in \mathbb{R}^{K \times d}$	Shared memory bank
τ	Memory retrieval temperature
α	Memory exponential-moving-average decay rate
p	Number of Laplacian eigenvectors in TopoPE
k	Number of random-walk steps in RWPE
$\mathbf{B}^{\text{base}}$	Base attention bias from adjacency and DSPE similarity
$\mathbf{B}^{\text{mem}}$	Memory-induced attention bias
$\mathbf{M}^{\text{mem}}$	Retrieved memory matrix
$\mathcal{A}_{\text{mask}}$	Attention mask over observed edges and self-loops
$\mathbf{h}^{\text{inv}}$	Domain-invariant representation
$\mathbf{h}^{\text{dom}}$	Domain-conditioned representation
$\mathcal{L}$	Total training objective
$\mathcal{L}_{GC}$	GC regression loss
$\mathcal{L}_{align}$	Domain alignment loss
$\mathcal{L}_{mem}$	Memory regularization loss
λ_{align}	Alignment-loss weight
λ_{mem}	Memory-regularization weight
δ	Huber-loss threshold parameter
q_k	Fraction of nodes assigned to memory slot k
R^2	Coefficient of determination (primary transfer metric)

B TRAINING AND EVALUATION PROTOCOLS

The source-only leave-one-domain-out (LODO) evaluation and supervised few-shot adaptation use different label-access and checkpoint-selection rules. They are specified separately below; Appendix D.2 gives the split and monitoring details. Neither procedure uses target-test labels to select a model.

Target-domain labels do not enter gradient updates, architecture or hyperparameter selection, early stopping, checkpoint selection, or seed/cross-run selection. The historical implementation also logged target metrics during training for diagnostics; these were not inputs to the monitor and were not consulted in any Table 1 selection decision. Algorithm 1 expresses the selection-isolated procedure, with target evaluation outside the training loop.

Algorithm 1 Source-only LODO training and target evaluation

Require: Source training sets S_{train}; source validation sets S_{val}; held-out target test set T_{test}; predetermined configuration and seeds
Ensure: Source-selected model and held-out target metrics

Set the target-training split to $\varnothing$
Initialize XRC-Former and the shared memory
Set the best source-validation MSE to $+\infty$
for each training epoch **do**
    for each source-domain training batch **do**
        Compute circuit encodings, memory retrieval, and GC predictions
        Compute GC loss on source training labels and alignment over source-domain pairs
        Optimize the training objective and update memory using source training data
    end for
    Compute aggregate MSE on S_{val} only
    if source-validation MSE improves **then**
        Save model parameters and memory as the best checkpoint
    end if
    Apply early stopping using source-validation MSE only
end for
Restore and freeze the best source-selected checkpoint
Evaluate the frozen checkpoint on T_{test}
Report all predetermined runs; do not select seeds or table entries by target metrics

Algorithm 2 Supervised few-shot target adaptation

Require: Source-selected checkpoint; labeled target support T_{sup}; disjoint labeled target validation T_{val}; fixed target test set T_{test}
Ensure: Adapted checkpoint and target-test metrics

Keep T_{sup}, T_{val}, and T_{test} mutually disjoint
Initialize the adaptation model from the source-selected checkpoint
for each adaptation epoch **do**
    Update the model using the permitted training data, including target-support labels
    Evaluate target-validation MSE on T_{val}, without validation-label gradient updates
    Select the best adaptation checkpoint and apply early stopping using target-validation MSE
end for
Restore and freeze the selected adaptation checkpoint
Evaluate on the fixed T_{test}

Algorithm 2 explicitly permits target supervision and does not describe the Table 1 zero-shot setting. The support-size axis counts support components only; target-validation labels, when used, are an additional supervision resource. A zero-target-label LODO reference follows Algorithm 1, not target-validation-based selection.

C Objective and Metric Details

This appendix collects the complete objective definitions and metric formulas used by the reported implementation. Appendix A summarizes the symbols shared with the main text, while the equations below provide implementation-level detail for training, evaluation, and memory updates.

$$\mathcal{L} = \mathcal{L}_{GC} + \lambda_{align}\mathcal{L}_{align} + \lambda_{mem}\mathcal{L}_{mem}. \tag{12}$$

GC regression loss. The GC term uses the Huber loss with threshold parameter δ:

$$\mathrm{Huber}_\delta(z) = \begin{cases} \frac{1}{2}z^2, & |z| \le \delta, \\ \delta\left(|z| - \frac{1}{2}\delta\right), & |z| > \delta. \end{cases} \tag{13}$$

For a training batch $\mathcal{B}$ of component graphs,

$$\mathcal{L}_{GC} = \frac{1}{|\mathcal{B}|}\sum_{G\in\mathcal{B}} \frac{1}{|\mathcal{M}_{\mathrm{net},G}|} \sum_{v\in\mathcal{M}_{\mathrm{net},G}} \mathrm{Huber}_\delta(\hat{c}_v - c_v). \tag{14}$$

MMD alignment loss. Alignment is applied to invariant representations from source-domain pairs using a Gaussian kernel:

$$\mathrm{MMD}^2_{\mathrm{Gauss}}(\mathbf{H}_i, \mathbf{H}_j) = \frac{1}{m_i^2}\sum_{a,b} k(\mathbf{h}_a, \mathbf{h}_b) + \frac{1}{m_j^2}\sum_{c,d} k(\mathbf{h}_c, \mathbf{h}_d) - \frac{2}{m_i m_j}\sum_{a,c} k(\mathbf{h}_a, \mathbf{h}_c), \tag{15}$$

where $k(\mathbf{x}, \mathbf{y}) = \exp(-\|\mathbf{x}-\mathbf{y}\|^2/2\sigma^2)$ and σ is chosen adaptively as the median pairwise distance of the combined set, clamped to $[10^{-3}, \infty)$. Each domain is subsampled to at most $m = 256$ nodes. The alignment objective used during training is

$$\mathcal{L}_{align} = \frac{1}{|\mathcal{P}|}\sum_{(i,j)\in\mathcal{P}} \mathrm{MMD}^2_{\mathrm{Gauss}}(\mathbf{H}_i^{\mathrm{inv}}, \mathbf{H}_j^{\mathrm{inv}}), \tag{16}$$

where $\mathcal{P}$ contains all source-domain pairs present in the current training fold.

Memory regularization. In the default `h_base` space,

$$\mathcal{L}_{mem} = \frac{1}{|\mathcal{M}_{\mathrm{net}}|}\sum_{v\in\mathcal{M}_{\mathrm{net}}} \|\mathbf{h}_v^{\mathrm{base}} - \mathbf{m}_v\|_2^2, \tag{17}$$

where $\mathbf{m}_v = \sum_k \beta_{vk}\mathbf{z}_k$ is the retrieved memory vector from the shared bank $\mathbf{Z} = \{\mathbf{z}_k\}_{k=1}^K$. In the optional `h_inv` space,

$$\mathcal{L}_{mem} = \frac{1}{|\mathcal{M}_{\mathrm{net}}|}\sum_{v\in\mathcal{M}_{\mathrm{net}}} \|\mathbf{h}_v^{\mathrm{inv}} - \mathbf{W}_{\mathrm{inv}}\mathbf{m}_v\|_2^2. \tag{18}$$

R^2 metric. Let $\bar{c} = \frac{1}{|\mathcal{M}_{\mathrm{net}}|}\sum_{v\in\mathcal{M}_{\mathrm{net}}} c_v$ denote the mean normalized GC label over the evaluated NET nodes. The reported coefficient of determination is

$$R^2 = 1 - \frac{\sum_{v\in\mathcal{M}_{\mathrm{net}}}(c_v - \hat{c}_v)^2}{\sum_{v\in\mathcal{M}_{\mathrm{net}}}(c_v - \bar{c})^2}. \tag{19}$$

Negative R^2 indicates worse performance than predicting the target-domain mean for every evaluated NET node.

Memory hard-assignment exponential moving average. After each training step, the memory bank is updated by hard assignment and slot-wise averaging. For each NET-node representation $\mathbf{h}_v^{\text{base}}$, define the assigned slot

$$\kappa_v = \arg\max_k \frac{(\mathbf{h}_v^{\text{base}})^\top \mathbf{z}_k}{\tau}. \tag{20}$$

For slot k, let $\mathcal{A}_k = \{v : \kappa_v = k\}$ and

$$\bar{\mathbf{h}}_k = \frac{1}{|\mathcal{A}_k|} \sum_{v \in \mathcal{A}_k} \mathbf{h}_v^{\text{base}}. \tag{21}$$

The slot update is then

$$\mathbf{z}_k \leftarrow \text{Normalize}\big(\alpha \mathbf{z}_k + (1-\alpha)\bar{\mathbf{h}}_k\big). \tag{22}$$

This hard-assignment exponential moving average stabilizes prototype evolution without differentiating through the retrieval softmax.

D Experimental Setup

This appendix collects the reproducibility details that support Section 5.1. We separate data construction, evaluation protocols, baseline adaptation, and repeated-run/hardware details so that each part of the experimental pipeline can be located directly.

D.1 Data Construction and Preprocessing

Circuit domains and technology platforms. The underlying circuit data are obtained from ParasGB (Zou et al., 2026) and organized into the component-level GC prediction and cross-domain evaluation setting described here. The benchmark contains three macro-domain corpora with a common typed-graph representation but different circuit families and technology platforms. Digital designs use the open-source Nangate45/FreePDK45 45-nm platform and Nangate Open Cell Library at the typical corner ($25\,^\circ\text{C}$, $1.1\,\text{V}$). The Digital designs share the same standard-cell library, routing stack, and extraction configuration. Analog cases use a 180-nm BCD process, while SRAM circuits use TSMC 28-nm CMOS. Thus, the present LODO benchmark includes both circuit-domain and technology/process differences; it does not isolate either factor as the sole cause of the observed transfer gap. A unified-PDK benchmark that better controls technology variation is ongoing work, not an evaluated result of this paper.

Graph representation and prediction unit. Each corpus is converted into typed circuit graphs with DEVICE, PIN, and NET nodes and explicit device–pin and pin–net connectivity. Digital graphs represent synthesized and routed IP blocks, Analog graphs represent analog circuit cases, and SRAM graphs represent memory arrays and peripheral circuitry. Here, *heterogeneous* refers to node and relation types, not to evaluation of a single integrated mixed-signal system. Supervision is restricted to NET-node ground capacitance. Raw-design statistics must be distinguished from model inputs: XRC-Former processes extracted local components with at most 256 nodes, rather than applying whole-design attention to the raw graphs.

Raw-design scale and scope. Six Digital designs exceed 100k raw graph nodes: `vga_enh_top` (553k), `swerv` (549k), `jpeg` (336k), `aes_xcrypt` (167k), `wb_conmax` (160k), and `aes_secworks` (131k). These counts describe the source corpora, not exhaustive full-design prediction coverage. Component-wise batching bounds the size of each model input; end-to-end full-design preprocessing cost, throughput, and industrial-flow validation have not been established here.

Reading the raw-statistics tables. Tables D.1–D.3 summarize the raw corpora before component extraction, truncation, and sampling. D–P and P–N denote device–pin and pin–net edge counts; k and M denote 10^3 and 10^6, respectively. The last three columns report the mean, minimum, and maximum supervised NET-node C_g in farads. Digital `c_ground_pf` labels are converted from pF to F for these statistics; Analog and SRAM raw labels are already in F. These physical-unit statistics are distinct from the normalized-label metrics used for model evaluation.

Table D.1: Digital raw circuit dataset statistics.

Dataset	Nodes				Edges			C_g statistics (F)		
	Total	Device	Pin	Net	Total	D–P	P–N	Mean	Min	Max
ss_pcm	2.48k	452	1.48k	551	2.93k	1.45k	1.48k	3.29E-16	1.42E-17	3.98E-15
ac97_ctrl	60.02k	10.79k	36.27k	12.96k	72.40k	36.13k	36.27k	7.54E-16	9.06E-18	9.83E-14
vga_enh_top	553k	101k	337k	116k	673k	336k	337k	2.53E-15	1.01E-17	1.20E-13
des3_area	9.82k	1.63k	6.35k	1.85k	12.50k	6.16k	6.35k	8.66E-16	1.32E-17	9.01E-15
des	14.27k	2.62k	8.66k	2.99k	17.12k	8.46k	8.66k	5.21E-16	1.13E-17	3.70E-14
aes	41.68k	7.58k	25.64k	8.46k	50.89k	25.25k	25.64k	8.36E-16	1.14E-17	1.38E-13
sha256	63.86k	11.39k	39.06k	13.41k	77.34k	38.28k	39.06k	9.46E-16	9.06E-18	3.12E-14
aes_secworks	131k	22.29k	83.40k	24.88k	166k	82.88k	83.40k	1.53E-15	8.61E-18	4.00E-14
aes_xcrypt	167k	28.43k	108k	30.20k	216k	108k	108k	2.03E-15	9.06E-18	8.04E-14
tv80	31.47k	5.55k	19.93k	5.99k	39.79k	19.86k	19.93k	9.33E-16	1.14E-17	1.16E-14
ibex	98.95k	17.06k	62.41k	19.47k	125k	62.15k	62.41k	1.05E-15	9.06E-18	3.74E-14
swerv	549k	93.68k	349k	106k	695k	347k	349k	1.78E-15	9.99E-18	5.15E-14
fifo	22.09k	3.62k	13.74k	4.73k	27.40k	13.66k	13.74k	8.88E-16	1.40E-17	2.29E-13
mem_ctrl	36.41k	6.30k	22.75k	7.36k	45.24k	22.48k	22.75k	8.09E-16	1.32E-17	7.36E-14
uart	2.73k	462	1.67k	596	3.28k	1.61k	1.67k	3.45E-16	1.65E-17	1.63E-14
sasc_top	2.79k	476	1.70k	609	3.37k	1.67k	1.70k	4.67E-16	1.74E-17	6.98E-15
simple_spi_top	3.73k	639	2.31k	786	4.59k	2.28k	2.31k	5.90E-16	1.32E-17	7.93E-15
spi_top	14.14k	2.43k	9.02k	2.69k	17.94k	8.93k	9.02k	9.71E-16	1.51E-17	5.72E-14
dynamic_node	70.19k	12.11k	43.44k	14.64k	86.19k	42.75k	43.44k	1.12E-15	7.75E-18	3.75E-14
pci-	74.24k	12.35k	46.31k	15.58k	92.25k	45.94k	46.31k	9.30E-16	1.25E-17	4.03E-14
usbf_top	59.83k	10.46k	36.65k	12.71k	73.06k	36.40k	36.65k	9.46E-16	1.14E-17	3.89E-14
wb_dma_top	16.64k	2.91k	10.15k	3.59k	19.86k	9.72k	10.15k	1.11E-15	1.25E-17	2.85E-14
wb_conmax	160k	27.70k	103k	29.46k	204k	101k	103k	3.67E-15	9.99E-18	3.18E-13
usb_phy	2.66k	462	1.61k	592	3.18k	1.57k	1.61k	4.62E-16	1.32E-17	5.88E-15
fir	31.05k	5.83k	18.52k	6.70k	36.98k	18.46k	18.52k	4.63E-16	1.09E-17	9.69E-14
jpeg	336k	62.28k	200k	73.98k	399k	199k	200k	8.81E-16	5.09E-18	7.21E-14
Total Average	98.27k	17.33k	61.05k	19.88k	122k	60.66k	61.05k	1.07E-15	1.15E-17	6.49E-14

Table D.2: Analog raw circuit dataset statistics.

Dataset	Nodes				Edges			C_g statistics (F)		
	Total	Device	Pin	Net	Total	D–P	P–N	Mean	Min	Max
case10	448	132	306	10	834	528	306	1.78E-13	1.41E-14	3.41E-13
case11	6.92k	1.74k	3.50k	1.68k	7.09k	3.60k	3.50k	2.18E-15	1.04E-15	2.76E-13
case15	1.84k	553	1.24k	41	3.41k	2.17k	1.24k	2.18E-13	1.04E-15	1.85E-12
case17	6.92k	1.74k	3.50k	1.68k	7.09k	3.60k	3.50k	2.18E-15	1.04E-15	2.76E-13
case1	1.10k	251	814	35	1.78k	962	814	5.22E-14	1.04E-15	3.01E-13
case23	1.57k	475	1.03k	64	2.82k	1.80k	1.03k	6.20E-14	1.06E-15	9.54E-13
case29	1.54k	478	1.05k	15	2.96k	1.91k	1.05k	2.56E-13	1.62E-14	1.22E-12
case39	1.30k	408	873	17	2.50k	1.63k	873	8.31E-14	2.81E-15	4.24E-13
case42	2.90k	902	1.98k	19	5.59k	3.61k	1.98k	3.03E-13	2.08E-14	1.01E-12
case44	1.41k	440	930	39	2.63k	1.70k	930	5.80E-14	1.04E-15	5.82E-13
case45	256	66	180	10	444	264	180	2.99E-14	5.45E-15	7.64E-14
case55	591	138	362	91	746	384	362	1.13E-14	1.03E-15	2.94E-13
case58	3.67k	1.02k	2.23k	417	5.54k	3.30k	2.23k	3.07E-14	1.03E-15	2.14E-12
case5	534	152	350	32	914	564	350	6.58E-14	1.07E-15	5.16E-13
case71	1.36k	413	924	27	2.55k	1.63k	924	1.65E-13	1.00E-15	8.08E-13
case72	1.45k	438	961	52	2.64k	1.68k	961	1.09E-13	1.04E-15	1.70E-12
case74	2.50k	758	1.63k	111	4.47k	2.84k	1.63k	6.64E-14	1.05E-15	1.35E-12
case75	231	68	156	7	428	272	156	2.70E-14	8.28E-15	6.61E-14
case78	2.39k	621	1.30k	463	2.87k	1.57k	1.30k	7.16E-15	1.04E-15	5.58E-13
case7	1.23k	382	800	43	2.26k	1.46k	800	5.60E-14	1.04E-15	8.97E-13
Total Average	2.01k	559	1.21k	243	2.98k	1.77k	1.21k	8.91E-14	4.11E-15	7.82E-13

Table D.3: SRAM raw circuit dataset statistics.

Dataset	Nodes				Edges			C_g statistics (F)		
	Total	Device	Pin	Net	Total	D–P	P–N	Mean	Min	Max
array_128_32_8t	144k	32.77k	98.30k	12.64k	197k	98.30k	98.30k	1.19E-16	6.66E-19	5.98E-14
digtime	17.58k	4.13k	12.40k	1.05k	24.80k	12.40k	12.40k	3.17E-17	1.14E-19	8.63E-15
sandwich	11.7M	2.65M	7.94M	1.15M	15.9M	7.94M	7.94M	2.46E-16	1.11E-25	9.10E-12
sp8192w	30.3M	6.98M	21.0M	2.34M	41.9M	21.0M	21.0M	8.07E-17	7.79E-22	7.51E-12
ssram	250k	57.42k	172k	19.91k	345k	172k	172k	5.48E-17	1.55E-25	1.43E-14
timing_ctrl	18.20k	4.38k	13.15k	668	26.30k	13.15k	13.15k	1.42E-16	3.31E-20	4.13E-15
ultra8t	10.2M	2.33M	6.97M	862k	13.9M	6.97M	6.97M	1.60E-16	1.41E-23	9.46E-12
Total Average	7.52M	1.72M	5.17M	628k	10.3M	5.17M	5.17M	1.19E-16	1.16E-19	3.74E-12

Node features $\mathbf{X} \in \mathbb{R}^{n \times d_{\text{in}}}$ contain graph-theoretic statistics, RC proxy features, routing-geometry indicators, and domain-specific preprocessing columns. Physics proxy features use indices $\{0, 3, 8, 11, 12, 13, 14, 15, 16\}$, and Digital geometry features use indices $\{8, \ldots, 15\}$. Edge types follow $\mathbf{e}_{uv} \in \{0, 1\}$ for $(u, v) \in E$ (0 = resistance, 1 = capacitance), while the dense edge-type matrix $\mathbf{E}^{\text{type}} \in \{0, 1, 2\}^{n \times n}$ uses 0 for no edge, 1 for resistance, and 2 for capacitance. Node types follow $\mathbf{t}_v \in \{0, 1, 2\}$ (0 = NET, 1 = DEVICE, 2 = PIN).

GC labels are normalized with `label_norm_mode = legacy`, i.e., the historical normalization used in the selected-XRC benchmark pipeline. All reported metrics are computed in this normalized label space without inverse transformation. The evaluation mask $\mathcal{M}_{\text{net}}$ restricts supervision and metric computation to NET nodes only.

D.2 Evaluation Protocols and Metrics

Source-only LODO (Table 1). For each held-out macro domain, the target-training split is empty. Source domains are partitioned by `split_source_train_val` with source-validation ratio 0.2. The target partitions are formed by `split_eval_only` with ratio 0.5; their existence does not permit target-label-based selection in LODO. The three folds are Digital+Analog→SRAM, Digital+SRAM→Analog, and Analog+SRAM→Digital. Model parameters are optimized exclusively using source-domain training labels. Aggregate source-validation MSE is the sole early-stopping and checkpoint-selection criterion. No target-domain labels are used for gradient updates, architecture or hyperparameter selection, seed selection, cross-run selection, or table-entry selection.

Checkpoint selection versus diagnostic evaluation. In the Table 1 runs, target-domain metrics were computed for diagnostic logging, but were neither passed to the checkpoint monitor nor consulted in any model-selection decision. Algorithm 1 makes the separation explicit: the training loop contains only source-domain training and validation; target evaluation is a separate step after the source-selected checkpoint is frozen. Target-domain training-dynamics curves are diagnostic analyses, not checkpoint-selection criteria.

Supervised few-shot adaptation (the few-shot experiments in Appendix E.2). Few-shot adaptation is a distinct, target-supervised protocol. The target-test partition remains fixed across support sizes. Labeled support components are drawn from the non-test target pool, and a disjoint target-validation split is reserved for adaptation checkpoint selection; the support, validation, and test components do not overlap. Support labels may be used for gradient updates, whereas target-validation labels may be used for early stopping and checkpoint selection only in this explicitly supervised setting. The reported support-size range is 0–5 target components. This is a support-component budget, not a count of all labels accessed when target validation is used. The source-only LODO selection rule must not be replaced by target validation merely because a run has zero support components. Algorithm 2 states the adaptation procedure separately from LODO.

The primary metric is the coefficient of determination R^2, defined in Appendix C. We additionally report mean squared error (MSE), mean absolute error (MAE), and root mean squared error (RMSE). All metrics are computed only over the evaluated NET nodes selected by $\mathcal{M}_{\text{net}}$. The target test set is excluded from gradient updates, early stopping, hyperparameter tuning, and checkpoint selection in both protocols.

D.3 Baselines and Hyperparameter Selection

All baselines are adapted to the same node-level GC regression task so that architectural differences rather than task mismatches dominate the comparison. The baseline families follow the same grouping used in Section 5 and Table 1: generic GNN backbones (GCN, GAT, GraphSAGE, PNA), graph Transformer backbones (SGFormer, PolyNormer), multi-domain graph models (MDGFM, BRIDGE), and circuit/parasitic predictors (MLParest, ParaGraph, ParaFormer, DLPL-Cap, CirGPS, CircuitGCL).

All methods share the same circuit graph schema, node features, edge indices, node-type labels, and GC supervision mask as XRC-Former. The common configuration uses hidden dimension 128, up to 3 layers where applicable, a 2-layer GC regression head, AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$,

weight decay 10^{-4}, learning rate 2×10^{-4}, at most 80 epochs, patience 15, and balanced batch sampling with $g_{\text{src}} = 4$ graphs per source domain.

Architecture-specific hyperparameters are tuned per baseline on source-domain validation. The search space covers number of layers $\{2, 3, 4\}$, hidden dimension $\{64, 128, 256\}$, learning rate $\{10^{-3}, 5 \times 10^{-4}, 2 \times 10^{-4}, 10^{-4}\}$, dropout $\{0.0, 0.1, 0.2, 0.3\}$, and family-specific settings such as attention heads for GAT, aggregator choices for PNA and GraphSAGE, and positional-encoding variants for graph Transformers. For XRC-Former, λ_{align} and λ_{mem} are fixed to 0.1 based on preliminary source-domain validation, and the remaining defaults are reported in Appendix F.

D.4 Repeated Runs and Hardware

Each experiment is repeated with multiple random seeds. The split seed defaults to the main seed (42) with deterministic per-domain offsets so that all methods use identical component partitions, while training seeds vary to capture initialization and optimization stochasticity. We report mean $\pm$ standard deviation for all metrics as variability summaries. Because the archived manuscript results are aggregated at the repeated-run level rather than stored as paired per-run tables for every method, we do not perform formal significance testing in this version of the paper.

All reported experiments use NVIDIA A100 GPUs with 40GB memory. XRC-Former and all baselines are evaluated on the same hardware class. CPU execution exists for debugging, but it is not used for reported results.

E Additional Results

This appendix collects supplementary empirical evidence that supports the experiments in Section 5. The results are organized by experimental role: supervised within-domain references, few-shot adaptation, additional training dynamics, additional ablations, extended error metrics, and hyperparameter sensitivity.

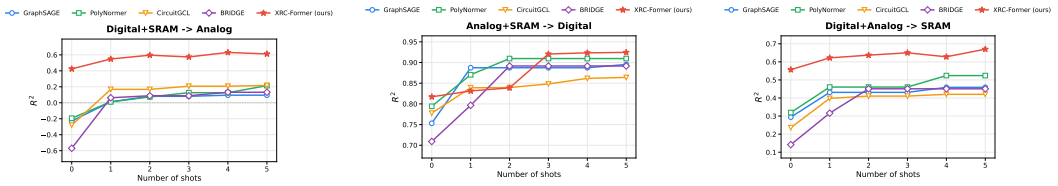
E.1 Supervised Within-domain Reference and Zero-shot Transfer

To contextualize zero-shot transfer accuracy, we additionally evaluate XRC-Former with supervised training and testing within each macro domain. The within-domain experiments use fixed train/validation/test splits, at most 80 training epochs, early-stopping patience 15, and five optimization seeds. Checkpoints are selected using the corresponding in-domain validation split. Because only one domain is used for training, no pair of distinct source domains exists: the alignment loss is inactive, with zero alignment pairs recorded in all 15 runs.

Table E1: XRC-Former under two supervision regimes. Entries are test R^2 (mean $\pm$ standard deviation); the supervised reference uses five seeds, and the zero-shot values reproduce Table 1. These are different training and model-selection protocols, not a comparison under equal target-label budgets.

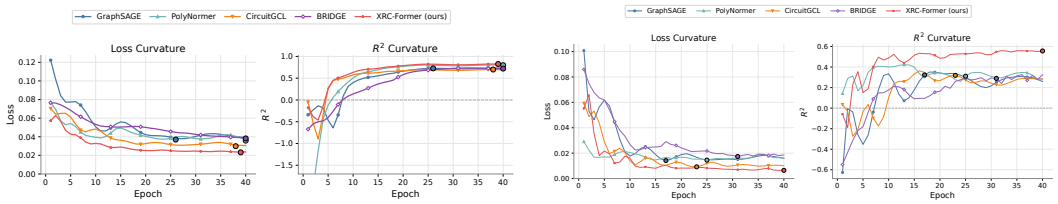
Evaluated domain	Source-only LODO	Supervised within-domain
Digital	0.8171 ± 0.0848	0.9760 ± 0.0056
Analog	0.4245 ± 0.1991	0.6739 ± 0.0635
SRAM	0.5567 ± 0.1002	0.7855 ± 0.0415
Target-label training	No	Yes
Checkpoint selection	Source validation	In-domain validation

The supervised models achieve higher mean R^2 in all three domains, providing reference points for the accuracy attainable with available in-domain supervision. They are not zero-shot competitors, an upper-bound guarantee, or direct evaluations of separate Analog-, Digital-, or SRAM-native prediction tools. The two protocols are not presented as a paired, identical-test-split comparison. Moreover, the within-domain results alone do not establish that active cross-domain alignment preserves source accuracy, since alignment is inactive in these runs.



(a) Analog target. (b) Digital target. (c) SRAM target.

Figure E1: Few-shot target-domain adaptation results for ground capacitance prediction across all three held-out domains. Models are first trained under the leave-one-domain-out split and then adapted with 0–5 labeled target-domain shots from the held-out domain. We report target-domain R^2; higher is better.



(a) Digital target. (b) SRAM target.

Figure E2: Additional training dynamics under source-only leave-one-domain-out transfer for the Digital and SRAM held-out targets. In each panel, the left plot reports source-validation MSE across epochs, while the right plot reports target-domain R^2 on labeled NET nodes. Higher target-domain R^2 indicates better cross-domain GC prediction.

The practical motivation for transfer is conditional on target-label availability. When representative target labels and a trained specialist already exist, supervised domain-specific training is a natural alternative. In the zero-target-label cold-start setting, such a new specialist cannot be trained from target labels; LODO instead reuses labeled source domains to provide predictions without target-domain gradient updates or model selection. Table E1 therefore illustrates the accuracy–supervision trade-off rather than universal superiority over specialists. End-to-end design-flow benefits and deployment-time savings have not been measured.

E.2 Few-shot Adaptation

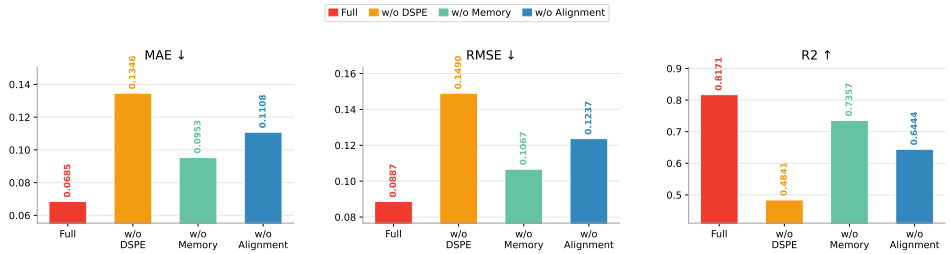
Figure E1 reports few-shot target adaptation curves for the Analog, Digital, and SRAM target domains in one grouped view. These results complement the main transfer evaluation by showing how performance changes once a small number of labeled target-domain support components becomes available while the target test set remains fixed.

E.3 Additional Training Dynamics

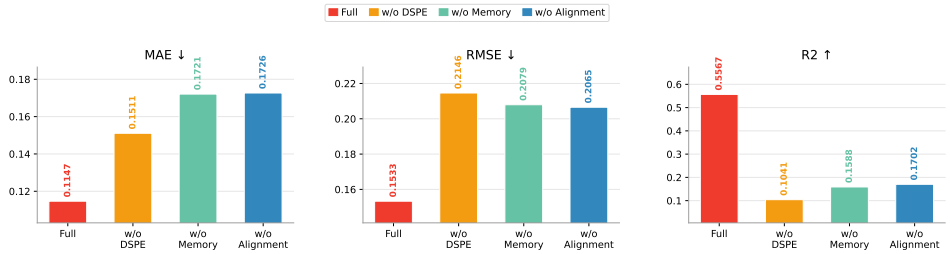
Figure E2 complements the main Analog-target training-dynamics result in Figure 4. It shows whether the relation between source-validation optimization and target-domain transfer behaves similarly when the held-out domain is Digital or SRAM.

E.4 Additional Ablations

Figure E3 complements the main Analog-target ablation in Figure 3. It tests whether the same transfer components remain useful when the held-out target is easier than Analog.



(a) Digital target.



(b) SRAM target.

Figure E3: Additional ablation studies for cross-domain ground capacitance prediction on the Digital and SRAM target domains. Removing DSPE, memory, or alignment degrades MAE, RMSE, and R^2 in both cases, showing that the proposed transfer components remain beneficial beyond the hardest Analog-target setting.

E.5 Extended Error Metrics

Tables E2 and E3 report extended target-test MAE and RMSE statistics for the same leave-one-domain-out setting as Table 1. We place these tables here so that the main paper can stay focused on target-domain R^2 while the appendix provides complementary absolute-error evidence.

Table E2: Extended cross-domain GC prediction results with target-test MAE. Lower is better. We report mean ± standard deviation over repeated runs. Best results are in bold and second-best results are underlined.

Category	Method	A+S→D	D+A→S	D+S→A
GNN	GCN (Kipf & Welling, 2017)	0.0949 ± 0.0387	0.1532 ± 0.0242	0.2138 ± 0.0060
	GAT (Veličković et al., 2018)	0.0963 ± 0.0266	0.1348 ± 0.0073	0.2097 ± 0.0074
	GraphSAGE (Hamilton et al., 2017)	0.0990 ± 0.0145	0.1334 ± 0.0078	0.2040 ± 0.0094
	PNA (Corso et al., 2020)	0.0850 ± 0.0237	0.1403 ± 0.0066	0.2068 ± 0.0093
Graph Transformer	SGFormer (Wu et al., 2023)	0.0791 ± 0.0122	0.1458 ± 0.0091	0.2123 ± 0.0172
	PolyNormer (Deng et al., 2024)	0.0815 ± 0.0131	0.1305 ± 0.0081	0.2053 ± 0.0089
Multi-Domain GFM	MDGFM (Wang et al., 2025)	0.1149 ± 0.0240	0.1546 ± 0.0060	0.2156 ± 0.0049
	BRIDGE (Yuan et al., 2025)	0.0786 ± 0.0488	0.1500 ± 0.0168	0.2187 ± 0.0266
Parasitic GNN	ParaFormer (Yoon et al., 2025)	0.0984 ± 0.0144	0.1330 ± 0.0049	0.2445 ± 0.0108
	MLParest (Shook et al., 2020)	0.1200 ± 0.0092	0.1778 ± 0.0033	0.2151 ± 0.0097
	DLPL-Cap (Shen et al., 2024)	0.1352 ± 0.0102	0.1781 ± 0.0016	0.1985 ± 0.0091
	ParaGraph (Ren et al., 2020)	0.0873 ± 0.0064	0.1220 ± 0.0021	0.2099 ± 0.0075
	CircuitGPS (Shen et al., 2025b)	**0.0666 ± 0.0148**	0.1320 ± 0.0059	0.2305 ± 0.0052
	CircuitGCL (Shen et al., 2025a)	0.0683 ± 0.0090	0.1331 ± 0.0016	0.1955 ± 0.0170
Ours	XRC-Former (ours)	0.0685 ± 0.0159	**0.1147 ± 0.0128**	**0.1306 ± 0.0170**

Table E3: Extended cross-domain GC prediction results with target-test RMSE. Lower is better. We report mean ± standard deviation over repeated runs. Best results are in bold and second-best results are underlined.

Category	Method	A+S→D	D+A→S	D+S→A
GNN	GCN (Kipf & Welling, 2017)	0.1088 ± 0.0398	0.2171 ± 0.0120	0.2605 ± 0.0057
	GAT (Veličković et al., 2018)	0.1130 ± 0.0265	0.1929 ± 0.0097	<u>0.2246 ± 0.0104</u>
	GraphSAGE (Hamilton et al., 2017)	0.1122 ± 0.0153	0.1905 ± 0.0046	0.2291 ± 0.0115
	PNA (Corso et al., 2020)	0.1031 ± 0.0238	0.2032 ± 0.0099	0.2359 ± 0.0120
Graph Transformer	SGFormer (Wu et al., 2023)	<u>0.0912 ± 0.0136</u>	0.2054 ± 0.0082	0.2375 ± 0.0229
	PolyNormer (Deng et al., 2024)	0.1046 ± 0.0091	0.1870 ± 0.0061	0.2259 ± 0.0056
Multi-Domain GFM	MDGFM (Wang et al., 2025)	0.1313 ± 0.0223	0.2141 ± 0.0061	0.2595 ± 0.0110
	BRIDGE (Yuan et al., 2025)	0.1034 ± 0.0457	0.2092 ± 0.0194	0.2555 ± 0.0464
Parasitic GNN	ParaFormer (Yoon et al., 2025)	0.1112 ± 0.0149	0.1948 ± 0.0052	0.2814 ± 0.0139
	MLParest (Shook et al., 2020)	0.1401 ± 0.0146	0.2303 ± 0.0033	0.2743 ± 0.0127
	DLPL-Cap (Shen et al., 2024)	0.1559 ± 0.0158	0.2253 ± 0.0026	0.2417 ± 0.0119
	ParaGraph (Ren et al., 2020)	0.1063 ± 0.0063	<u>0.1821 ± 0.0090</u>	0.2465 ± 0.0122
	CirGPS (Shen et al., 2025b)	0.1013 ± 0.0067	0.1899 ± 0.0035	0.2632 ± 0.0117
	CircuitGCL (Shen et al., 2025a)	0.0978 ± 0.0100	0.1982 ± 0.0042	0.2340 ± 0.0318
Ours	XRC-Former (ours)	**0.0887 ± 0.0151**	**0.1533 ± 0.0171**	**0.1631 ± 0.0274**

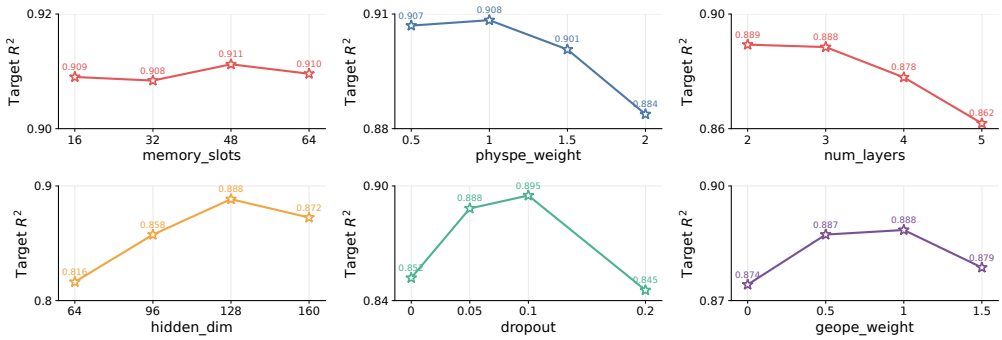


Figure E4: Hyperparameter sensitivity for node-level cross-domain ground capacitance prediction on the Digital held-out target domain. We vary memory size, physics-encoding weight, model depth, hidden dimension, dropout, and geometry-encoding weight, and report target-domain R^2. The results show that XRC-Former is relatively stable over moderate hyperparameter ranges, while overly large physics weights, excessive depth, or excessive dropout can degrade transfer performance.

E.6 Hyperparameter Sensitivity

This analysis evaluates sensitivity to memory size, physics-encoding weight, Transformer depth, hidden dimension, dropout, and geometry-encoding weight in the Digital-target setting. Figure E4 shows two observations. ① **XRC-Former is locally stable around the default operating point.** Moderate changes to memory size, hidden dimension, geometry weight, and regularization yield similar transfer performance, suggesting that the gains do not come from a narrowly tuned single configuration. ② **This stability is local rather than universal.** Excessive physics weight, depth, or dropout can degrade transfer, so the result should be interpreted as benchmark-specific stability around a reasonable operating region rather than universal robustness.

F Implementation Details

This appendix summarizes the implementation-level details needed to reproduce the reported XRC-Former results. Appendix D covers data construction, evaluation protocols, baseline adaptation, repeated runs, and hardware; the present section focuses on the default model configuration, optional architecture variants, prediction-head details, and implementation-specific notes.

Default configuration. Unless otherwise specified, XRC-Former uses the default profile rather than the expanded `draft_arch` profile. Table F1 lists the complete default hyperparameter specification.

Table F1: Complete default XRC-Former hyperparameters.

Module	Parameter	Value
Model	Hidden dimension d	128
	Transformer layers L	3 (2 shared + 1 final)
	Attention heads h	4
	Dropout rate	0.1
Memory	Memory slots K	64
	Temperature τ	0.2
	EMA decay α	0.99
	Init mode	random
	Regularization space	`h_base`
TopoPE	Laplacian eigenvectors p	8
	RWPE steps k	4
	Shared MLP	2-layer, GELU, hidden 128
GeoPE	Mode	`stable`
	Feature columns	$\{8, 9, 10, 11, 12, 13, 14, 15\}$
	Per-domain MLP	2-layer, GELU, hidden 128
	Zero-fallback domains	Analog, SRAM
PhysPE	Mode	`proxy`
	Feature columns	$\{0, 3, 8, 11, 12, 13, 14, 15, 16\}$
	Additional features	log-degree, log edge-type counts
	Per-domain MLP	2-layer, GELU, hidden 128
	NET-only standardization	disabled
Attention	Mode	`masked` (graph-neighborhood)
	Structural bias mode	`adj_plus_pos`
	Adjacency bias β_{adj}	0.15
	Positional bias β_{pos}	0.35
	Edge-type attention bias	disabled
Optional modules	PNA local branch	disabled (weight = 0.25)
	Degree scalers (if PNA enabled)	enabled
	Type-specific input proj	disabled
	Domain token	disabled
	Learnable physics gate	disabled
Loss	Huber delta δ	0.05
	λ_{align}	0.1
	λ_{mem}	0.1
	MMD bandwidth σ	adaptive median
	Max alignment nodes	256
Training	Optimizer	AdamW
	β_1, β_2	0.9, 0.999
	Weight decay	10^{-4}
	Learning rate η_0	2×10^{-4} (constant)
	LR schedule	`none` (no warmup, no decay)
	Epochs	80
	Early stopping patience	15
	Monitor metric	source val MSE (LODO)
	Monitor smoothing window	1 epoch
Data	Graphs per source domain g_{src}	4
	Graphs per target epoch g_{tgt}	4
	Label normalization	`legacy`
Reproducibility	Random seed	42
	Split seed	defaults to main seed

Draft architecture configuration. The expanded `draft_arch` profile is designed for larger capacity and longer training. When `-draft_arch 1` is specified, parameters whose current values match parser defaults are overwritten as listed in Table F2.

Table F2: Draft architecture configuration overrides relative to the default profile.

Parameter	Default Value	Draft Value
Hidden dimension d	128	256
Transformer layers L	3	6
Memory slots K	64	256
TopoPE eigenvectors p	8	16
RWPE steps k	4	8
Huber delta δ	0.05	1.0
Learning rate η_0	2×10^{-4}	10^{-4}
LR schedule	`none` (constant)	`cosine`
Warmup epochs	0	10
Total epochs	80	200
Early stopping patience	15	20
GeoPE mode	`stable`	`draft`
PhysPE mode	`proxy`	`draft`
Attention mode	`masked`	`global`
Structural bias mode	`adj_plus_pos`	`pos_only`
Memory init mode	`random`	`kmeans_first_epoch`
Memory reg space	`h_base`	`h_inv`
Edge-type attention bias	disabled	enabled

Parameters not listed above retain their default values in the draft profile. The draft configuration is not used in the main experiments unless explicitly stated.

Prediction head. The final prediction head consists of two linear projections that decompose the final hidden state into domain-invariant and domain-conditioned components, followed by a 2-layer regression MLP:

$$\hat{c}_v = \mathbf{w}_2^\top \, \mathrm{GELU}(\mathbf{W}_1(\mathbf{h}_v^{\mathrm{inv}} + \mathbf{h}_v^{\mathrm{dom}}) + \mathbf{b}_1) + b_2. \tag{23}$$

Here $\mathbf{W}_1 \in \mathbb{R}^{d \times d}$ and $\mathbf{w}_2 \in \mathbb{R}^d$, with dropout 0.1 between the two layers. Predictions are evaluated only on NET nodes against the GC label c_v.

Implementation diagnostics. During training, we log the number of active slots (prototypes with at least one assignment), assignment entropy $H = -\sum_k q_k \log q_k$, and the top-slot assignment share. These diagnostics are used only for monitoring and are not part of the optimization objective.

Baseline implementation notes. Baselines are implemented within the `run_model_suite` framework using the same graph construction and data loading pipeline as XRC-Former. Each baseline receives the same circuit graph, the same GC supervision mask $\mathcal{M}_{\mathrm{net}}$, the same deterministic data splits, and a comparable GC prediction head for fair architectural comparison.

ParaFormer, MLParest, DLPL-Cap, ParaGraph, CirGPS, and CircuitGCL are adapted from their original implementations to the node-level GC regression task studied here. Their original coupling-capacitance heads, resistance heads, multi-task settings, or classification objectives are replaced with a standard GC regression head while preserving the core architectural contribution of each method.

Anonymous code and reproducibility artifact. The anonymized implementation and experimental configurations are available at:

`https://anonymous.4open.science/r/review-artifact-7f3c-43E3/`

The artifact includes XRC-Former training and evaluation code, preprocessing and split logic, configurations for LODO, few-shot adaptation, supervised within-domain evaluation, and ablations, together with random seeds, environment specifications, checkpoint-selection rules, and reproduction commands. The code release should be distinguished from access to the underlying circuit data; it does not imply that all raw circuit designs and technology-dependent data are publicly redistributable.

Additional anonymized files are available through the OSF view-only link:

Anonymous OSF files (view-only)

G Additional Limitations

This appendix collects the main scope boundaries and deployment caveats so that the main text can remain focused on the evaluated transfer results.

Task scope. The main experiments focus on node-level ground capacitance prediction on heterogeneous circuit component graphs. This is a controlled entry point for studying cross-domain parasitic representation transfer, but it does not by itself establish performance on broader parasitic tasks.

Data and deployment. The reported results cover three macro domains—Digital, Analog, and SRAM—and rely on proprietary circuit design data and extraction outputs. Practical deployment would require broader validation across technology nodes, design families, and production flows, and reproducibility remains partly dependent on access to suitable proprietary or future open data.

Broader parasitic quantities. Coupling capacitance, resistance prediction, full resistance-capacitance (RC) extraction, standard parasitic exchange format (SPEF) generation, and signoff replacement are outside the evaluated scope of this paper. These settings introduce additional supervision structure, physical constraints, and workflow requirements beyond the present study.

H Additional Literature Context

This appendix records additional literature that informs the broader context of cross-domain graph representation learning, while not being central to the main experimental comparison.

Graph foundation models and graph pre-training build on broader foundation-model and self-supervised learning ideas (Bommasani et al., 2021; Hu et al., 2020b; Sun et al., 2020; Lu et al., 2021; Liu et al., 2023a; Xie et al., 2023; Cao et al., 2023; Xu et al., 2023).

Related work on graph transfer, prompting, and zero-shot or cross-domain graph generalization further motivates our transfer setting (Zhu et al., 2021; Zhang et al., 2022; Sun et al., 2022; Liu et al., 2023b; Zhao et al., 2024; Li et al., 2024; Wu et al., 2022; You et al., 2023). We also follow standard concerns from scalable graph learning and GNN evaluation when designing repeated-run and split-controlled experiments (Zeng et al., 2020; Shchur et al., 2018).