This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

R²G : A Multi-View Circuit Graph Benchmark Suite from RTL to GDSII

Zewei Zhou[1], Jiajun Zou[1], Jiajia Zhang[1], Ao Yang[1], Ruichao He[1], Haozheng Zhou[1]
Ao Liu[1], Jiawei Liu[2], Leilei Jin[2], Shan Shen[1,*], Daying Sun[1,*]
[1]Nanjing University of Science and Technology, [2]The Chinese University of Hong Kong
{zhouzewei, shanshen, hasdysun}@njust.edu.cn

Abstract

Graph neural networks (GNNs) are increasingly applied to physical design tasks such as congestion prediction and wirelength estimation, yet progress is hindered by inconsistent circuit representations and the absence of controlled evaluation protocols. We present ***R2G*** *(RTL-to-GDSII), a multi-view circuit-graph benchmark suite that standardizes* five stage-aware views *with* information parity *(every view encodes the same attribute set, differing only in where features attach) over 30 open-source IP cores (up to* 10^6 *nodes/edges). R2G provides an end-to-end DEF-to-graph pipeline spanning synthesis, placement, and routing stages, together with loaders, unified splits, domain metrics, and reproducible baselines. By decoupling representation choice from model choice, R2G isolates a confound that prior EDA and graph-ML benchmarks leave uncontrolled. In systematic studies with GINE, GAT, and ResGatedGCN, we find: (i) view choice dominates model choice, with Test* R^2 *varying by more than 0.3 across representations for a fixed GNN; (ii) node-centric views generalize best across both placement and routing; and (iii) decoder-head depth (3–4 layers) is the primary accuracy driver, turning divergent training into near-perfect predictions (*R^2*>0.99). Code and datasets:* https://github.com/ShenShan123/R2G.

1. Introduction

AI for Electronic Design Automation (EDA) is gaining rapid attention. A growing range of physical design tasks, including placement, routing, timing, and congestion prediction, can be tackled by deep learning models [15, 19, 25, 35]. Modern circuit layouts are structured spatial scenes where components occupy two-dimensional coordinates and interact through connectivity relations. Yet circuit graphs carry structural complexity absent from general-purpose graph datasets: the graph ML community has built rich resources for citation networks [11], social graphs [17], molecular property prediction [24, 31], and biological networks [37], and has matured around benchmarks such as OGB and TUDataset [9, 13, 21], yet these datasets encode neither typed heterogeneous entities and multi-terminal hyperedges nor geometry-aware attributes. Circuit graphs contain entities with explicit types and hierarchy whose coordinates and orientations are tightly coupled with electrical constraints, spanning multiple non-interchangeable physical design (PD) stages. More critically, the same circuit admits multiple legitimate representations in late physical design, including netlist graphs, spatial graphs, and hypergraphs, whose preserved spatial semantics and message-passing paths differ fundamentally.

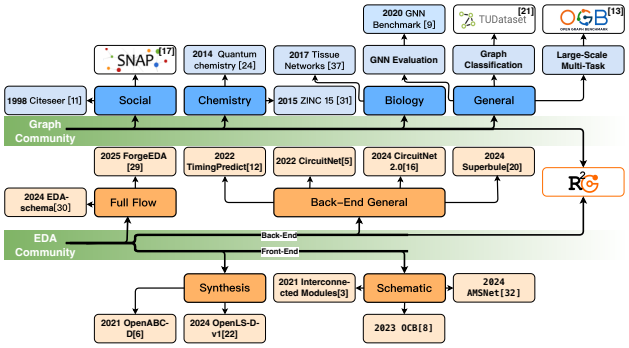

Figure 1. Benchmark and dataset evolution across graph ML and EDA, highlighting R²G's unique position at their intersection as a stage-aware, multi-view circuit-graph benchmark suite.

Despite mature open-source flows and growing graph-oriented resources enabling scalable data generation [1, 2], existing resources cannot support systematic study of graph representation. Front-end datasets are fixed to a single logical representation such as AIG or Boolean circuits, emphasizing logic structure over back-end spatial semantics [6, 22]; schematic-level resources focus on netlist understanding and analog topology, remaining distant from digital late-PD tasks [3, 8, 32]. Back-end datasets co-determine graph representation and task-specific labels, so view choice has never been studied as an independent variable [5, 12, 16, 20]. Cross-stage resources have advanced in flow coverage, yet are not designed for the controlled com-

*Corresponding authors

parison of fixing circuits and tasks while varying only the graph view [29, 30]. The result is that views, stages, and tasks are deeply entangled, and the independent effect of graph representation on performance remains indiscernible: it is impossible to determine whether a model's accuracy reflects its architecture or its choice of graph representation. Figure 1 situates this gap within the broader evolution of graph ML and EDA datasets.

To address this gap, we propose **R2G** (**R**TL-**to**-**G**DSII), which standardizes *five stage-aware views* with *information parity* over 30 IP cores, extracting graphs, features, and labels directly from DEF with loaders, unified splits, and reproducible baselines. Our contributions are:

- **Multi-view circuit-graph benchmark:** Five complementary views of the *same* late-PD circuits with information parity, stage-aware features, and labels for node-level placement (HPWL) and edge-level routing (wire length), enabling controlled study of representation effects absent from prior single-view EDA datasets.
- **Cross-view findings:** For a fixed GNN, Test R^2 varies by more than 0.3 across views and model rankings flip, showing that representation choice dominates model selection. Node-centric views are most robust; view (b) performs best across both stages.
- **Decoder-head depth as a dominant lever:** Head depth (3–4 layers) matters far more than GNN depth: raising head layers from 1 to 4 lifts Test R^2 from -0.17 to 0.99 for placement and resolves NaN divergence for routing [9, 13].
- **Open-source pipeline:** End-to-end DEF-to-graph pipeline, loaders, unified splits, and baselines over datasets spanning up to 10^6 nodes/edges. Usage examples: Appendix A.

2. Related Work

R2G sits at the intersection of graph-ML benchmarking and EDA datasets; we discuss each in turn and then articulate the gap both communities have left open. A detailed comparison table appears in Appendix B.

2.1. Benchmarks in the Graph ML Field

Graph ML benchmarking has matured around large-scale, standardized datasets. OGB [13] introduced unified splits and metrics across node, link, and graph tasks, and TUDataset [21] consolidated graph classification collections. Datasets for social and citation networks [7, 11, 17], molecules [9, 24, 31], and proteins [13, 37] established best practices for loaders, baselines, and evaluation. Despite these advances, all of these datasets are domain-agnostic and predominantly single-view: they encode neither typed heterogeneity and multi-terminal hyperedges nor geometry-aware attributes required for physical design, and none offers multiple representations of the *same* data to isolate representation effects.

Circuit graphs differ fundamentally from these benchmarks: a single design reaches 10^4–10^6 nodes and edges in incidence-style views, while geometry- or routing-resource views add $\mathcal{O}(W \times H \times L)$ edges (where W, H, and L denote layout width, height, and metal-layer count); DEF files provide explicit 2D layout cues (cell coordinates, orientations); and both topology and spatial structure are intrinsic to the representation.

2.2. Benchmarks in the EDA Field

Open EDA datasets have advanced ML for circuits across stages and tasks. Front-end datasets such as OpenABC [6] and OpenLS-D-v1 [22] are built on fixed logical representations like AIG or Boolean circuits, emphasizing synthesis over back-end spatial semantics. Schematic-level resources including GNN-RE [3], CktGNN [8], and AMSNet [32] address netlist understanding and analog topology outside the digital late-PD scope. Back-end datasets including CircuitNet [5], CircuitNet2.0 [16], the TimingPredict dataset [12], and Superblue [20] target congestion, timing, and routability tasks, but co-determine graph representation and labels—so view choice has never been studied as an independent variable. Cross-stage resources such as EDA-schema [30] and ForgeEDA [29] have advanced flow coverage and multimodal aggregation, alongside EDALearn [23] and synthetic generators [14], yet none is designed for controlled cross-view comparison under fixed circuits and tasks.

Yet the field still lacks *standard graph benchmarks*: existing resources adopt a single representation per task or stage with inconsistent node/edge types, features, and label resolutions, and no unified conversion from design files to typed graphs with information parity across stages. The key gap is therefore benchmark formulation rather than data availability. R2G addresses this by inheriting graph-ML best practices (unified splits, scalable loaders, reproducible baselines) while providing typed, multi-view circuit graphs with information parity and resolution-matched supervision, offering the first controlled framework for evaluating circuit graph representations across PD stages.

3. R2G Benchmark Construction

We now detail how R2G is constructed, beginning with the design corpus and its graph-relevant characteristics.

3.1. Digital Designs

We begin with the R2G design corpus: 30 open-source IP cores from OpenCores and GitHub spanning processors, DSP, cryptography, communication/system control, and video/audio control. Table 1 summarizes gate and net counts, top-level netlist/DEF IO counts, node and edge

Table 1. Design corpus statistics including circuit sizes and corresponding graph scale (#Nodes and #Edges under view (b)) for representative open-source IP cores.

Designs	Category	Circuit & Graph Statistics					Function
		#Gates	#Nets	#IOs	#Nodes	#Edges	
ss_pcm	Audio controller	463	521	28	2.44k	2.88k	pulse code modulation
ac97_ctrl	Audio controller	10509	12531	132	58.22k	70.60k	AC97 Audio Codec Controller
vga_lcd	Video controller	97688	110587	198	531.33k	650.84k	VGA/LCD Display Controller
des3_area	Crypto core	1529	1655	190	9.78k	12.46k	3DES Enc/Dec Module
systemcdes	Crypto core	2663	2788	197	14.17k	17.02k	SystemC DES Encryption/Decryption Core
systemcaes	Crypto core	7565	7963	389	41.08k	50.29k	SystemC AES Encryption/Decryption Core
sha256	Crypto core	10637	12461	774	62.83k	76.32k	SHA256 Hash Accelerator
aes_secworks	Crypto core	21700	23941	520	128.40k	164.11k	Area-optimized AES Implementation
aes_xcrypt	Crypto core	28337	29595	391	165.76k	215.07k	AES Engine with Enc/Dec Support
tv80	Processor	5634	5880	63	31.14k	39.45k	Z80 Compatible Processor Core
tv80s	Processor	7298	7377	46	37.24k	45.74k	Z80 Compatible Processor Core
riscv32i	Processor	9712	10446	135	53.23k	66.91k	Simplified RISC Variant Processor
ibex	Processor	16523	18760	264	96.60k	122.18k	Ibex 32-bit RISC-V CPU Core
tinyRocket	Processor	27770	32239	269	50.63k	36.27k	TinyRocket RISC-V Processor Core
swerv	Processor	89893	102155	2039	538.36k	684.89k	SweRV EH1 RISC-V Core
bp_multi	Processor	158629	139704	1453	296.72k	250.15k	Black Parrot multi-core processor
uart	Communication controller	478	550	54	2.73k	3.29k	Universal Async Receiver/Transmitter
sasc_top	Communication controller	696	726	28	3.45k	4.17k	Simple Async Serial Comm Controller
i2c_verilog	Communication controller	883	959	33	4.69k	5.70k	Inter-Integrated Circuit Bus Controller
simple_spi_top	Communication controller	936	968	28	4.57k	5.49k	Simple SPI Controller
spi_top	Communication controller	3031	3050	92	15.62k	19.18k	SPI Controller
dynamic_node	Communication controller	11196	13560	693	67.87k	83.87k	Dynamic Network Router Node
pci	Communication controller	12041	14915	369	72.69k	90.70k	PCI Interface Controller
usb_funct	Communication controller	12222	14121	249	67.30k	81.58k	USB Function Controller
wb_dma_top	System controller	3416	3840	432	19.52k	23.03k	WISHBONE DMA Controller
wb_conmax	Communication controller	32161	33312	2546	181.45k	221.76k	WISHBONE Connection Matrix
usb_phy	Communication controller	555	588	33	2.73k	3.22k	USB Physical Layer
fir	DSP core	4077	4777	66	22.63k	27.68k	Finite Impulse Response Filter
jpeg	DSP core	61865	73108	47	331.18k	394.38k	JPEG Encoder/Decoder
idft	DSP core	138312	178202	132	817.88k	1009.40k	Inverse Discrete Fourier Transform

counts, and each design's function and category. For discussion, we consolidate these 30 designs into five broad categories and highlight their graph-relevant traits.

- **Video/Audio controllers:** Structured dataflow with modest IO counts.
- **Communication/System controllers:** Protocol-centric controllers and bus interconnects emphasize handshakes, arbitration, and high external connectivity; graphs highlight interface nets, control trees, and contention points.
- **Crypto cores:** Dense combinational logic with staged datapaths; regular rounds stress arithmetic-heavy paths.
- **DSP cores:** Arithmetic throughput and wide datapaths; structured pipelines and repeated arithmetic motifs.

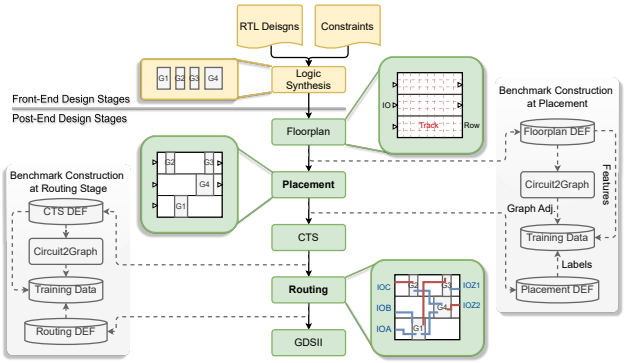

Figure 2. OpenROAD RTL-to-GDSII post-end flow including floorplanning, placement (global and detailed), clock-tree synthesis, routing (global and detailed), and signoff. R2G focuses on placement and routing, while earlier stages provide constraints and contextual information.

- **Processors:** Largest circuit and graph scales; diverse fan-in/fan-out; mixed control–datapath structure.

Across categories, circuit/graph scale, interface complexity, and topology vary, motivating careful *view selection* and *task formulation*. Compute-dense cores emphasize spatial coordination and placement signals, whereas interface-heavy controllers expose stronger routing, fanout, and congestion patterns.

3.2. Post-End Flow

Physical design transforms RTL into manufacturable layouts through the RTL-to-GDSII post-end flow. Following the OpenROAD pipeline (Figure 2), the flow includes synthesis, floorplanning, macro placement, global and detailed placement, clock tree synthesis (CTS), global and detailed routing, and signoff verification [2]. DEF files are produced after floorplanning, placement, and routing, together with routing guides and timing or congestion reports, encoding geometric layout, circuit topology, and electrical information that provide the basis for graph construction. A fragment of the DEF template is shown in Appendix C.

R2G focuses on the placement and routing stages because they largely determine timing, power consumption, and routability. Supervision is derived from physically meaningful quantities produced by the flow and is attached to whichever entity instantiates the target in a given graph view. For placement, these quantities are derived from placed cell geometry and net bounding boxes; for routing, they describe realized interconnect properties such as routed wire length and via usage.

Placement quality is evaluated using half-perimeter wirelength (HPWL), derived from bounding boxes of connected gates. Routing quality is measured using the exact routed wirelength after metal layer assignment and via insertion. These labels remain consistent across stages: placement HPWL approximates routing effort, while final routed

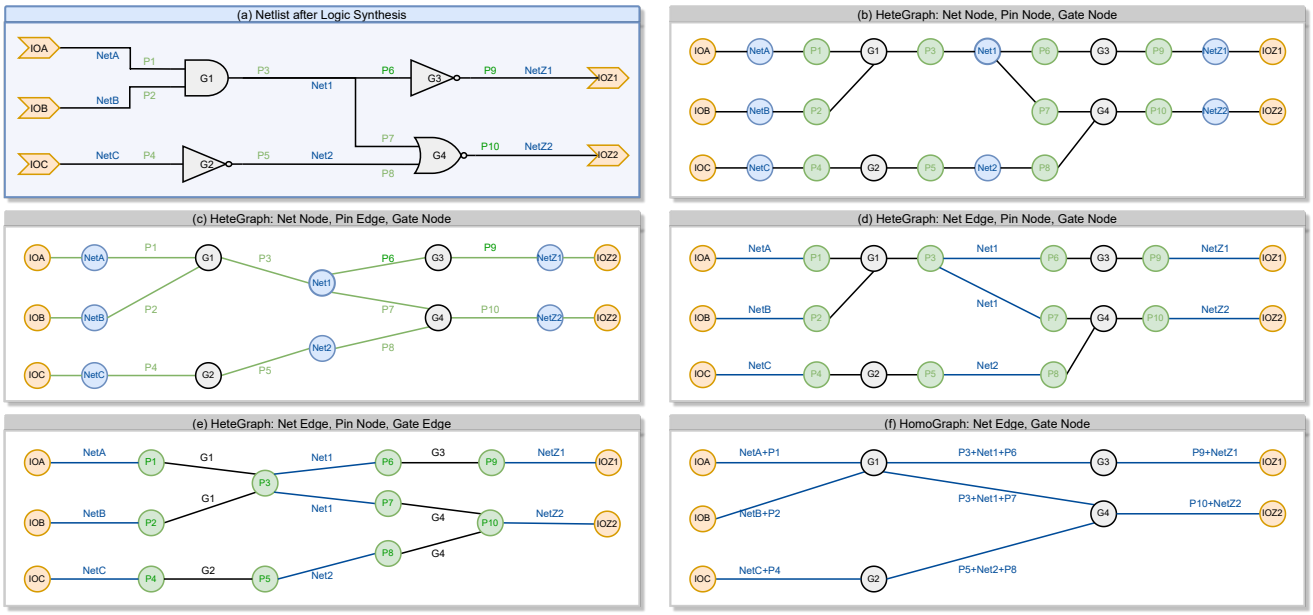

Figure 3. Original circuit schematic (a) and five complementary circuit graph views (b–f) used in R2G: (b) all-elements-as-nodes, (c) pins-as-edges, (d) nets-as-edges, (e) net–gate incidence edges, (f) net edges without pin nodes. These views support placement and routing analyses under different supervision granularities.

wirelength reflects realized interconnect across metal layers.

Existing datasets often isolate individual steps such as synthesis trajectories or single-stage layout snapshots [5, 6, 16, 23], leading to fragmented graph representations and evaluation protocols. In contrast, R2G captures the full placement and routing stages and tracks structural changes directly from DEF, including topology updates (e.g., buffering and clock-tree sink insertion) and feature shifts related to placement density, geometric constraints, routing layers, and via usage. Outputs from earlier stages, such as floorplan constraints, CTS sinks, and power grid structures, are incorporated as contextual features rather than prediction targets. This design preserves the modular OpenROAD flow while grounding multi-view circuit graphs in consistent stage-aware supervision.

3.3. Multi-View Circuit Graphs

Key graph definitions. We represent *digital* circuits as typed, attributed graphs $G = (\mathcal{V}, \mathcal{E})$ whose nodes and edges carry domain features drawn from DEF. A *heterogeneous* circuit graph includes multiple node and edge types to encode EDA semantics faithfully. R2G is built from four fundamental entity types: (i) *logic gates*, (ii) directional *pins* (input/output/inout), (iii) electrical *nets* connecting multiple pins (multi-terminal connectivity), and (iv) *IOs* (macro ports and pads). Crucially, the same entity may appear as a node or an edge depending on the view: a net becomes a node in bipartite or hypergraph views to capture multi-terminal connectivity, but an edge in pairwise views; pins can similarly be explicit nodes or encoded as incidence attributes on gate–net edges.

We provide an original circuit schematic (a) and five complementary views (b–f) in Figure 3, each panel tied to a consistent conversion rule and emphasizing distinct logical, geometric, and resource aspects of the same circuit:

a) **Original circuit schematic:** gate-level functional diagram with signals and IOs prior to graph conversion; serves as the source representation.
b) **All-elements-as-nodes:** gates, pins, nets, and IOs are nodes; typed edges encode relations (connects-to, drives, belongs-to). This maximizes semantic coverage.
c) **Pins-as-edges:** pins become typed edges between gate and net nodes; preserves incidence and direction while reducing size.
d) **Nets-as-edges:** nets become edges between gate nodes; emphasizes pairwise connectivity.
e) **Net–Gate incidence-as-edges:** net–gate membership is encoded with edges in a bipartite formulation; retains multi-terminal semantics and scales to large designs.
f) **Net edges without pin nodes:** pin nodes are pruned; nets remain as edges between gates.

Task alignment and empirical usage. Researchers select views to match signals and labels rather than a fixed front-/post-end dichotomy. View (b) supports early-stage coupling capacitance prediction [26]. View (c) is used for early-stage ground capacitance prediction [25] and for placement-stage congestion and `HPWL` prediction [36]. View (d) emphasizes pairwise connectivity and is adopted

Table 2. Definitions of features/labels for circuit components in terms of views.

Unit	Name	Type	Definition	Notes
Gate	x	feat. (rout.)	Gate x (DBU)	Node (b/c/d/f); edge (e)
Gate	y	feat. (rout.)	Gate y (DBU)	Node (b/c/d/f); edge (e)
Gate	cell_type	feat.	Integer encoding of gate type	Node (b/c/d/f); edge (e)
Gate	orientation	feat.	Orientation code (0–7)	Node (b/c/d/f); edge (e)
Gate	area	feat.	Gate area (tech lib)	Node (b/c/d/f); edge (e)
Gate	place_flag	feat.	Placement status	Node (b/c/d/f); edge (e)
Gate	power_leak	feat.	Leakage power (tech lib)	Node (b/c/d/f); edge (e)
Net	net_type	feat.	Net category encoding	Node (b/c); edge (d/e/f)
Net	pin_count	feat.	Pin count	Node (b/c); edge (d/e/f)
Net	HPWL	label (place.)/ feat. (rout.)	HPWL (DBU) from pin bbox	Node (b/c); edge (d/e/f)
Net	wire_length	label (rout.)	Exact routed wire length (DBU), per-layer Manhattan	node (b/c), edge (d/e/f)
Net	via_count	label (rout.)	Via count	Node (b/c); edge (d/e/f)
IO	x	feat.	IO x (DBU)	Node (b/c/d/e/f)
IO	y	feat.	IO y (DBU)	Node (b/c/d/e/f)
IO	orientation	feat.	IO orientation	Node (b/c/d/e/f)
IO	layer_id	feat.	Metal layer index	Node (b/c/d/e/f)
Pin	pin_type	feat.	Pin type encoding	Node (b); edge (c/e); aggregated (d/f)
Pin	cell_type	feat.	Owning gate type encoding	Node (b); edge (c/e); aggregated (d/f)

for analog circuit topology generation [10]. View (e) encodes net–gate membership suitable for timing-graph net delay prediction at the placement stage [16]. View (f) prunes pin nodes for scalability and is widely used for functional and synthesis tasks, including truth table prediction and SAT solving [27, 28], circuit synthesis [6], and logic synthesis with PPA prediction [29].

View choices across stages. The appropriate *inductive bias* follows the task: capacitance estimation benefits from direction and membership in view (b/c); placement metrics (congestion, HPWL) and timing rely on geometry attached to gates/edges in view (c/e); analog topology generation leverages pairwise net connectivity in view (d); functional and synthesis tasks favor scalable net-edge view (f). This diversity underscores a fidelity–scalability trade-off and motivates standardized, resolution-matched evaluation across views. More statistical information of five views is compared in Appendix D.

R2G standardization. R2G provides unified DEF-to-graph conversion across five typed, attributed views (b–f), enabling resolution-matched evaluation for placement and routing and controlled cross-view ablations that disentangle connectivity versus geometry signals. Selecting view per task avoids forcing a single representation and supports transfer across designs and technology nodes.

3.4. Feature & Label Extraction

3.4.1. Feature collection and assembly

We standardize feature definitions for four unit types (gates, nets, IOs, pins) and align them to the multi-view circuit graphs in Figure 3, views (b–f). Table 2 specifies the schema and sources (DEF and technology libraries). To maintain a coherent narrative, we first describe stage-specific feature sets and then detail how they are attached under each view.

Feature sets are stage-specific. During placement, gates omit x/y and rely on static attributes such as cell_type, area, and power_leak. During routing, gates include placed x/y, orientation, and place_flag. Nets include net_type, pin_count, and HPWL as a derived feature once gate positions are determined. IO features comprise x/y, orientation, and layer_id. Pins expose two categorical entries: pin_type encodes the pin type, and cell_type encodes the owning gate type.Coordinates (x, y) are reported in DBU (Database Unit).

To ensure reproducibility and consistent ordering across designs and views, discrete fields use stable integer vocabularies: cell_type maps logic-gate categories to IDs (e.g., 0–95); orientation maps to codes 0–7; net_type encodes {signal, power, ground, clock, reset, scan}; layer_id indexes metal layers; and place_flag encodes placement status.

With features defined, we keep feature parity across views: the same set of attributes is available in every view—no fields are missing and none are duplicated across views. We vary only attachment location to enable fair comparison across Figure 3, while preserving information content: (b) All-elements-as-nodes keeps gate, pin, net, and IO features on their respective nodes; (c) Pins-as-edges moves pin features to gate–net incidence edges while gates and nets retain node features; (d) Nets-as-edges attaches net features to pairwise gate–gate edges and propagates them deterministically to edges derived from the same net (without introducing new attributes); (e) Net–Gate incidence-as-edges attaches gate features to incidence edges between net and gate nodes to preserve bipartite semantics; and (f) Net edges without pin nodes concatenates pin and net features into a single attribute on gate–gate net edges. When multiple pins exist between a gate pair, pin features are aggregated with a fixed reducer before concatenation to avoid duplicating information. This design ensures feature parity across views and isolates representation effects.

3.4.2. Label calculation

We compute task-specific net labels consistently across stages. For placement, supervision uses HPWL, defined as the minimal bounding box over a net's pins:

$$\texttt{HPWL} = (x_{\max} - x_{\min}) + (y_{\max} - y_{\min}). \quad (1)$$

After placement, HPWL becomes a derived routing feature because gate positions are available. For routing, supervision uses exact wire_length from DEF routed geometry.

For each metal layer ℓ, there is

$$\texttt{wire_length} = \sum_{\ell \in \mathcal{L}} \sum_{(x_1,y_1),(x_2,y_2)\in \ell} |x_2 - x_1| + |y_2 - y_1|. \tag{2}$$

Here, $\mathcal{L}$ denotes the routed metal layers parsed from the nets section. We also parse an auxiliary routing label, `via_count`, from DEF tokens.

Consistent with the chosen representation, labels attach to the entity that represents the net: when a net is a node (views b/c in Figure 3), the label attaches to that node; when a net is an edge (views d–f), the label attaches to that edge. As designs progress through physical stages, the supervisory target changes accordingly: `HPWL` supervises placement and then transitions to a derived feature after placement, whereas `wire_length` and `via_count` supervise routing. This convention preserves stage-aware supervision while informing downstream views with physically grounded features.

4. Experiments

We report three standard regression metrics: mean absolute error (MAE), root mean squared error (RMSE), and coefficient of determination (R^2).

To enable fair cross-view comparison, we convert typed, heterogeneous graph views into homogeneous graphs with unified node and edge types while preserving attribute parity across views. This allows classic homogeneous GNNs, GINE [34], GAT [33], and ResGatedGCN [4], to serve as baselines across all views. In practice, homogeneous GNNs are widely adopted in EDA learning systems for scalability and lower engineering complexity, making them suitable for controlled benchmark comparisons. The experimental setup can be found in Appendix E. We leave heterogeneous representations and hetero-GNNs to future work.

4.1. Comparison of Different Views for Placement Tasks

Metrics follow MAE/RMSE/R^2; lower MAE/RMSE and higher R^2 indicate better performance. Train/Validation/Test splits are merged into a single wide table for side-by-side comparison (Table 3). The Avg. Test R^2 row reports the mean across the three GNN baselines, providing a view-level summary. Intuitively, node-centric graphs align more directly with gate-position labels than edge-only formulations or pin-incidence encodings, helping explain the stronger generalization of view (b). Critically, node-centric views (b/c) are preferred by all classic GNNs, consistently surpassing edge-only views (d/e) and the widely used AIG-equivalent view (f).

Across views, view (b) is consistently strongest on Test R^2, with view (c) close for ResGatedGCN. View (d) is mid-tier; view (e) collapses for GINE/ResGatedGCN; and view (f) is moderate. These trends are mirrored by MAE/RMSE, indicating practical significance beyond variance. Importantly, view (b) fits the physical design stages well and outperforms view (f) by large margins, suggesting view (f) should not be the default representation for learning-based placement.

Across models, performance is view-dependent: on view (b), GINE slightly edges ResGatedGCN; on view (c), ResGatedGCN surpasses GINE; on view (d), GINE $>$ ResGatedGCN; on view (e), both GINE and ResGatedGCN are negative whereas GAT is barely positive; on view (f), ResGatedGCN $>$ GINE $\gg$ GAT. The pattern suggests that residual gating stabilizes edge-aware message passing on node-centric graphs, while attention-only GAT underperforms in geometry-coupled regimes. Our *critical finding* lies here: for a fixed GNN, view choice affects performance far more than expected; rankings turn over across views. For example, GAT achieves its best accuracy only on view (e).

Negative R^2 on view (e) indicates a label–view mismatch: incidence-edge supervision dilutes spatial inductive bias needed for node targets. Moderate performance on view (f) suggests it can serve as auxiliary supervision rather than a primary view. This pattern underscores the importance of matching representation to target geometry.

With placement trends established, we next examine whether these view-dependent patterns persist when labels are connectivity-driven.

4.2. Comparison of Different Views for Routing Tasks

Routing is connectivity-driven; consequently, incidence and net-edge formulations (views (e) and (d)) should be competitive, while view (b) remains strong due to coupled geometry and features. Using the same models and metrics, Table 4 reports Train/Validation/Test performance across views (b–f).

Despite connectivity-centric labels, the node-centric view (b) remains the strongest across baselines, reinforcing the general preference for node-centric representations.

Across views, view (b) shows the best generalization on the test split. Among edge-based views, view (e) performs strongest overall, with view (d) remaining competitive, while view (f) consistently underperforms. These trends are consistent across data splits.

Across models, architectures exhibit clear view preferences. ResGatedGCN performs best on node-centric views (b,c), GINE remains competitive across views (b–d), and GAT performs best on view (e) but degrades on other views.

Our *critical finding* is that view choice dominates model choice: model rankings change substantially across graph representations. Attention-based models perform well on view (e) but lag on views (b–d), while GINE and ResGat-

Table 3. Placement results across views (b–f) and splits. Lower MAE/RMSE is better; higher R^2 is better.

		(b)			(c)			(d)			(e)			(f)		
Split	**Model**	**MAE ↓**	**RMSE ↓**	**R^2 ↑**	**MAE ↓**	**RMSE ↓**	**R^2 ↑**	**MAE ↓**	**RMSE ↓**	**R^2 ↑**	**MAE ↓**	**RMSE ↓**	**R^2 ↑**	**MAE ↓**	**RMSE ↓**	**R^2 ↑**
Train	GINE	**0.2859**	**0.6117**	**0.9554**	0.5061	0.8678	0.9017	0.3272	0.4516	0.7984	**0.1649**	**0.2936**	**0.9137**	**0.3334**	**0.4563**	**0.7941**
	ResGatedGCN	0.3097	0.655	0.9489	**0.4808**	**0.798**	**0.9169**	**0.3263**	**0.4477**	**0.8016**	0.1819	0.3508	0.8767	0.3339	0.4609	0.7938
	GAT	0.4762	1.1623	0.839	1.0883	1.8336	0.5611	0.4834	0.6394	0.5958	0.1812	0.3374	0.8861	0.6848	0.8683	0.2541
Validation	GINE	**0.6332**	**1.2972**	**0.8219**	1.095	1.6276	0.6929	**0.3604**	**0.4516**	**0.748**	0.4134	0.7032	0.5137	0.5431	0.6576	0.4674
	ResGatedGCN	0.7485	1.4158	0.7879	**0.8544**	**1.2884**	**0.8075**	0.3772	0.4809	0.7142	**0.3637**	**0.5815**	**0.6675**	**0.4462**	**0.5689**	**0.6108**
	GAT	1.1039	2.2889	0.4455	1.9788	2.3872	0.3393	0.4518	0.5596	0.6131	0.4071	0.7779	0.4049	0.7117	0.8487	0.2643
Test	GINE	**0.3468**	**1.0331**	**0.8878**	0.7194	1.3095	0.8003	**0.4337**	**0.5804**	**0.7109**	0.4755	0.9655	-0.0107	0.4903	0.6615	0.6151
	ResGatedGCN	0.3694	1.0668	0.8803	**0.6355**	**1.1277**	**0.8519**	0.4576	0.6043	0.6866	0.4930	0.9892	-0.0610	**0.4717**	**0.6207**	**0.7067**
	GAT	0.5068	1.3621	0.8049	1.6859	2.6390	0.1889	0.6109	0.7568	0.5085	**0.4276**	**0.9374**	**0.0472**	0.8857	1.0320	0.0634
Avg. Test		**0.4077**	**1.1540**	**0.8577**	1.0136	1.6921	0.6137	0.5007	0.6472	0.6353	0.4654	0.9640	−0.0082	0.6159	0.7714	0.4617

Table 4. Routing results across views (b–f) and splits. Lower MAE/RMSE is better; higher R^2 is better.

		(b)			(c)			(d)			(e)			(f)		
Split	**Model**	**MAE ↓**	**RMSE ↓**	**R^2 ↑**	**MAE ↓**	**RMSE ↓**	**R^2 ↑**	**MAE ↓**	**RMSE ↓**	**R^2 ↑**	**MAE ↓**	**RMSE ↓**	**R^2 ↑**	**MAE ↓**	**RMSE ↓**	**R^2 ↑**
Train	GINE	**0.2520**	0.7796	0.9257	0.7011	1.3372	0.8716	0.4019	0.5419	0.7105	0.4188	0.5744	0.6746	**0.1477**	**0.2130**	**0.9551**
	ResGatedGCN	0.3041	**0.6848**	**0.9427**	**0.5291**	**0.9252**	**0.9385**	**0.3947**	**0.5400**	**0.7126**	0.3805	0.5217	0.7318	0.5545	0.7204	0.4886
	GAT	0.3944	1.0904	0.8547	1.4430	2.5930	0.5170	0.5659	0.7216	0.4871	**0.2914**	**0.4246**	**0.8224**	0.8544	1.0354	-0.0618
Validation	GINE	0.8523	1.7891	0.6529	1.7403	2.8566	0.4877	**0.3892**	**0.5013**	**0.6601**	0.3839	0.5155	**0.6370**	**0.2670**	**0.4098**	**0.8143**
	ResGatedGCN	**0.7971**	**1.5267**	**0.7472**	**1.0107**	**1.7078**	**0.8169**	0.4222	0.5288	0.6218	0.3929	0.5237	0.6255	0.6243	0.7871	0.1746
	GAT	1.1039	2.2035	0.4734	2.6724	3.3680	0.2879	0.5718	0.7148	0.3089	**0.1054**	**0.3233**	0.4816	0.8053	0.9564	-0.0111
Test	GINE	**0.2994**	1.1373	0.8596	1.2648	2.3667	0.6471	**0.5746**	**0.7391**	**0.5758**	0.6993	0.8588	0.4277	0.8106	0.9531	0.2972
	ResGatedGCN	0.3498	**1.0676**	**0.8763**	**0.7856**	**1.4808**	**0.8618**	0.5912	0.7640	0.5467	0.5001	0.6548	0.6673	**0.7634**	**0.9402**	**0.3179**
	GAT	0.4477	1.3123	0.8131	2.2493	3.7245	0.1260	0.8769	1.1125	0.0389	**0.3202**	**0.4557**	**0.8389**	1.0043	1.1361	0.0014
Avg. Test		**0.3656**	**1.1724**	**0.8497**	1.4332	2.5240	0.5450	0.6809	0.8719	0.3871	0.5065	0.6564	0.6446	0.8594	1.0098	0.2055

edGCN alternate as the strongest models depending on the representation.

The strong performance of GAT on view (e) suggests that attention benefits from incidence-edge neighborhoods capturing dense local connectivity, whereas pairwise nets in view (d) create sparser structures that weaken attention aggregation. ResGatedGCN remains stable on views (b,c), likely due to residual-gated aggregation that preserves geometric coupling. Generalization gaps widen on edge-only views, indicating weaker alignment between connectivity-only inputs and routing labels.

Compared with placement, routing is generally more challenging for classical GNNs, with lower predictive accuracy and larger generalization gaps.

Taken together, view (b) is the most reliable representation for routing tasks, while view (e) serves as the strongest edge-oriented complement. View (f) should generally be avoided. Model selection should match the representation: ResGatedGCN is most reliable on views (b,c), whereas GAT can be effective on view (e) but is less stable across views.

4.3. Comparison of GNNs and Task-Specific Heads

We now turn from view-level recommendations to model capacity: how message-passing depth and decoder-head design drive performance.

Building on these findings, on node-centric view (b), a moderate message-passing depth (3–4 layers) suffices; deeper stacks risk over-smoothing and optimization decay [18]. More importantly, *decoder-head configuration* is the dominant lever for placement and routing accuracy, a dimension underexplored in graph ML/EDA benchmarks that primarily tune the message-passing stack [4, 9, 13, 34]. With information parity across multi-view, typed circuit graphs and stage- and resolution-matched targets, R2G reveals that increasing head depth from 1 to 3–4 layers transforms performance and training stability (often flipping model rankings). With this context, we first summarize the *layer* ablations below, then return to *head* effects.

Table 5. Placement on view (b) vs. GNN depth (3–6). Compact stacks (3–4) yield highest accuracy; deeper (6) underperform.

	GINE				ResGatedGCN			
#layers	**MAE↓**	**RMSE↓**	**R^2↑**	**#Param.**	**MAE↓**	**RMSE↓**	**R^2↑**	**#Param.**
3	0.3489	1.0454	0.8851	0.78 M	0.3319	**1.0296**	**0.8885**	0.98 M
4	**0.2594**	**1.0158**	**0.8915**	0.98 M	0.3249	1.0617	0.8815	1.24 M
5	0.2991	1.0445	0.8853	1.18 M	**0.2923**	1.0699	0.8797	1.51 M
6	0.3312	1.2536	0.8348	1.38 M	0.3327	1.0833	0.8766	1.77 M

Table 6. Routing on view (b) vs. GNN depth (3–6). Performance peaks at 3 layers for both architectures; deeper stacks show diminishing returns.

	GINE				ResGatedGCN			
#layers	**MAE↓**	**RMSE↓**	**R^2↑**	**#Param.**	**MAE↓**	**RMSE↓**	**R^2↑**	**#Param.**
3	0.3220	**1.0691**	**0.8759**	0.78 M	0.3439	**1.0682**	**0.8762**	0.98 M
4	0.3466	1.1503	0.8664	0.98 M	**0.2977**	1.1141	0.8653	1.24 M
5	**0.2845**	1.0969	0.8694	1.18 M	0.3188	1.0768	0.8742	1.51 M
6	0.3670	1.0910	0.8708	1.38 M	0.3892	1.1495	0.8566	1.77 M

4.3.1. GNN Layers vs. Accuracy

For placement on view (b), accuracy varies non-monotonically with depth (Table 5). Both GINE and ResGatedGCN achieve their best performance at moderate

Table 7. Placement on view (b) vs. head layers (1–4). Increasing head capacity stabilizes training and drives accuracy; 3–4 head layers deliver near-perfect R^2 across architectures.

	GINE				ResGatedGCN			
#layers	MAE↓	RMSE↓	R^2↑	#Param.	MAE↓	RMSE↓	R^2↑	#Param.
1	0.4708	1.1886	0.8514	0.91 M	3.1580	3.3342	-0.1689	1.18 M
2	0.3414	1.1184	0.8685	0.98 M	0.3249	1.0542	0.8832	1.24 M
3	0.0870	0.3329	0.9883	1.05 M	**0.0963**	0.3251	0.9889	1.31 M
4	**0.0703**	**0.1909**	**0.9962**	1.11 M	0.1421	**0.3223**	**0.9891**	1.38 M

Table 8. Routing on view (b) vs. head layers (1–4). Shallow heads are unstable; accuracy improves sharply at 3 and is near-perfect at 4. Prefer 3–4 head layers for robust routing.

	GINE				ResGatedGCN			
#layers	MAE↓	RMSE↓	R^2↑	#Param.	MAE↓	RMSE↓	R^2↑	#Param.
1	13.9757	54.1022	nan	0.91 M	nan	nan	nan	1.18 M
2	0.3425	1.1414	0.8586	0.98 M	0.2892	1.1276	0.8620	1.24 M
3	0.0761	0.3173	0.9891	1.05 M	**0.1230**	**0.3823**	**0.9841**	1.31 M
4	**0.0542**	**0.1802**	**0.9965**	1.11 M	0.2482	0.3824	**0.9841**	1.38 M

depths and decline as depth increases, indicating diminishing returns.

Moderate depth balances receptive-field coverage and optimization stability, whereas deeper stacks risk over-smoothing and gradient decay. The sharper degradation observed for GINE suggests greater depth sensitivity of its aggregation mechanism. Consequently, compact stacks (3–4 layers) are preferred for node-centric placement on view (b).

Turning to routing (Table 6), we observe a similar non-monotonic pattern: both GINE and ResGatedGCN perform best at moderate depths and degrade as depth increases. Excessive depth diffuses signal across nets and increases training variance; the drop observed for ResGatedGCN at intermediate depths may indicate gating saturation. Accordingly, compact stacks (3–4 layers) are also preferred for routing on view (b).

4.3.2. Head Layers vs. Accuracy

On placement, performance rises sharply with head depth: moving from head 1 to head 3 lifts Test R^2 from 0.8514 to 0.9883 (GINE) and from -0.1689 to 0.9889 (ResGatedGCN); head 4 reaches near-perfect accuracy for GINE (0.9962) and maintains or slightly improves ResGatedGCN (0.9891). Moreover, shallow heads cannot linearize the mapping from node features to coordinates; additional depth provides the nonlinearity and normalization capacity needed for stable generalization. The negative R^2 at head 1 for ResGatedGCN is a clear outlier and reflects under-parameterization and poor calibration. Consequently, use 3–4 head layers on view (b); this yields stable, near-perfect placement accuracy.

For routing, head 1 is unstable (`nan` metrics); accuracy improves sharply at head 3 and reaches near-perfect at head 4 (GINE 0.9965; ResGatedGCN 0.9841). Furthermore, insufficient head capacity causes optimization failure and numerical instability; nan metrics at head 1 confirm this. Routing labels encode local connectivity patterns requiring nonlinear fusion, which shallow heads cannot provide. Accordingly, adopt 3–4 head layers for routing on view (b); shallow heads are brittle and should be avoided.

Overall, on node-centric view (b), compact message-passing depth (3–4 layers) suffices, while deeper stacks show over-smoothing and diminishing returns [18]. In contrast, *decoder-head capacity* is the primary driver of accuracy and stability: increasing the head from shallow settings to 3–4 layers consistently turns unstable optimization, particularly for routing, into accurate predictions. This pattern holds across architectures and can flip model rankings depending on view and head choices. Supplementary experiments on view (d) show the same qualitative behavior for edge-centric settings, as reported in Appendix F.

R2G's uniqueness lies in its controlled, multi-view, stage- and resolution-matched design with information parity, which decouples representation from modeling and makes decoder-head effects visible in ways that prior benchmarks, which emphasize message-passing stacks, seldom do [13, 34]. Practically, pair view (b) with 3–4 GNN layers and 3–4 head layers for robust, high accuracy, and prioritize view/head configuration before increasing depth.

5. Conclusions & Future Work

We have presented R2G, the first multi-view circuit-graph benchmark suite for physical design, providing five stage-aware views with information parity over 30 open-source IP cores, together with an end-to-end DEF-to-graph pipeline, loaders, unified splits, domain metrics, and reproducible baselines. Experiments with classic GNNs yield three transferable findings: (i) view choice dominates model choice, with Test R^2 varying by more than 0.3 across representations for a fixed GNN; (ii) node-centric views generalize best across both placement and routing, with effective views aligning with supervision granularity; and (iii) decoder-head depth (3–4 layers) is the primary accuracy driver, turning divergent training into near-perfect predictions (R^2>0.99), a finding prior benchmarks have not surfaced. For practitioners, we recommend view (b) as the default representation, 3–4 GNN layers to avoid over-smoothing, and prioritizing decoder-head configuration before increasing model depth.

Limitations and future work. R2G currently covers late physical-design stages; DEF-derived attributes omit timing, IR-drop, and detailed power; designs span limited technology nodes; and evaluations center on classic GNNs. Future work includes broader design and technology coverage, timing- and congestion-aware tasks with richer LEF/STA/PDN attributes, graph transformers and MoE architectures, and stronger EDA evaluation protocols for broader adoption.

References

[1] Openroad documentation. https://openroad.readthedocs.io/en/latest/main/README.html, . Accessed: 2025-11-06. 1

[2] Openroad: Unified application for rtl-to-gdsii flow. https://github.com/The-OpenROAD-Project/OpenROAD, . Accessed: 2025-11-06. 1, 3

[3] Lilas Alrahis, Abhrajit Sengupta, Johann Knechtel, Satwik Patnaik, Hani Saleh, Baker Mohammad, Mahmoud Al-Qutayri, and Ozgur Sinanoglu. GNN-RE: Graph neural networks for reverse engineering of gate-level netlists. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(8): 2435–2448, 2022. 1, 2

[4] Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017. 6, 7

[5] Zhuomin Chai, Yuxiang Zhao, Yibo Lin, Wei Liu, Runsheng Wang, and Ru Huang. Circuitnet: An open-source dataset for machine learning applications in electronic design automation (eda). *arXiv preprint arXiv:2208.01040*, 2022. 1, 2, 4

[6] Animesh Basak Chowdhury, Benjamin Tan, Ramesh Karri, and Siddharth Garg. Openabc-d: A large-scale dataset for machine learning guided integrated circuit synthesis. *arXiv preprint arXiv:2110.11292*, 2021. 1, 2, 4, 5

[7] Graph Consulting. The cora dataset. https://graphsandnetworks.com/the-cora-dataset/. Accessed: 2025-11-06. 2

[8] Zehao Dong, Weidong Cao, Muhan Zhang, Dacheng Tao, Yixin Chen, and Xuan Zhang. Cktgnn: Circuit graph neural network for electronic design automation. In *The Eleventh International Conference on Learning Representations*, 2023. 1, 2

[9] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020. 1, 2, 7

[10] Jian Gao, Weidong Cao, Junyi Yang, and Xuan Zhang. Analoggenie: A generative engine for automatic discovery of analog circuit topologies. *arXiv preprint arXiv:2503.00205*, 2025. 5

[11] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98, 1998. 1, 2

[12] Zizheng Guo, Mingjie Liu, Jiaqi Gu, Shuhan Zhang, David Z Pan, and Yibo Lin. A timing engine inspired graph neural network model for pre-routing slack prediction. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1207–1212, 2022. 1, 2

[13] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020. 1, 2, 7, 8

[14] Michael D Hutton, Jonathan Rose, Jerry P Grossman, and Derek G Corneil. Characterization and parameterized generation of synthetic combinational benchmark circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(10):985–996, 2002. 2

[15] Xun Jiang, Zizheng Guo, Zhuomin Chai, Yuxiang Zhao, Yibo Lin, Runsheng Wang, and Ru Huang. Accelerating routability and timing optimization with open-source ai4eda dataset circuitnet and heterogeneous platforms. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2023. 1

[16] Xun Jiang, Yuxiang Zhao, Yibo Lin, Runsheng Wang, Ru Huang, et al. Circuitnet 2.0: An advanced dataset for promoting machine learning innovations in realistic chip design environment. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2, 4, 5

[17] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, 2014. 1, 2

[18] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018. 7, 8

[19] Peiyu Liao, Dawei Guo, Zizheng Guo, Siting Liu, Yibo Lin, and Bei Yu. Dreamplace 4.0: Timing-driven placement with momentum-based net weighting and lagrangian-based refinement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(10):3374–3387, 2023. 1

[20] Zhishang Luo, Truong Son Hy, Puoya Tabaghi, Michaël Defferrard, Elahe Rezaei, Ryan M Carey, Rhett Davis, Rajeev Jain, and Yusu Wang. DE-HNN: An effective neural model for circuit netlist representation. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, pages 4258–4266, 2024. 1, 2

[21] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020. 1, 2

[22] Liwei Ni, Rui Wang, Miao Liu, Xingyu Meng, Xiaoze Lin, Junfeng Liu, Guojie Luo, Zhufei Chu, Weikang Qian, Xiaoyan Yang, Biwei Xie, Xingquan

Li, and Huawei Li. Openls-dgf: An adaptive open-source dataset generation framework for machine learning tasks in logic synthesis. *arXiv preprint arXiv:2411.09422*, 2024. 1, 2

[23] Jingyu Pan, Chen-Chia Chang, Zhiyao Xie, Yiran Chen, and Hai Helen Li. Edalearn: A comprehensive rtl-to-signoff eda benchmark for democratized and reproducible ml for eda research. *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, 2023. 2, 4

[24] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134k molecules. *Scientific Data*, 1:140022, 2014. 1, 2

[25] 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*, pages 440–445, 2024. 1, 4

[26] 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)*, pages 1–7. IEEE, 2025. 4

[27] Zhengyuan Shi, Hongyang Pan, Sadaf Khan, Min Li, Yi Liu, Junhua Huang, Hui-Ling Zhen, Mingxuan Yuan, Zhufei Chu, and Qiang Xu. Deepgate2: Functionality-aware circuit representation learning. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2023. 5

[28] Zhengyuan Shi, Ziyang Zheng, Sadaf Khan, Jianyuan Zhong, Min Li, and Qiang Xu. Deepgate3: Towards scalable circuit representation learning. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2024. 5

[29] Zhengyuan Shi, Zeju Li, Chengyu Ma, Yunhao Zhou, Ziyang Zheng, Jiawei Liu, Hongyang Pan, Lingfeng Zhou, Kezhi Li, Jiaying Zhu, et al. Forgeeda: A comprehensive multimodal dataset for advancing eda. *arXiv preprint arXiv:2505.02016*, 2025. 2, 5

[30] Pratik Shrestha, Alec Aversa, Saran Phatharodom, and Ioannis Savidis. Eda-schema: A graph datamodel schema and open dataset for digital design automation. In *Proceedings of the Great Lakes Symposium on VLSI 2024*, pages 69–77, 2024. 2

[31] Teague Sterling and John J. Irwin. Zinc 15—ligand discovery for everyone. *Journal of Chemical Information and Modeling*, 55(11):2324–2337, 2015. 1, 2

[32] Zhuofu Tao, Yichen Shi, Yiru Huo, Rui Ye, Zonghang Li, Li Huang, Chen Wu, Na Bai, Zhiping Yu, Ting-Jung Lin, and Lei He. Amsnet: Netlist dataset for ams circuits. In *2024 IEEE LLM Aided Design Workshop (LAD)*, pages 1–5. IEEE, 2024. 1, 2

[33] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 6

[34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018. 6, 7, 8

[35] Dingcheng Yang, Haoyuan Li, Wenjian Yu, Yuanbo Guo, and Wenjie Liang. Cnn-cap: Effective convolutional neural network-based capacitance models for interconnect capacitance extraction. *ACM Transactions on Design Automation of Electronic Systems*, 28 (4):1–22, 2023. 1

[36] Shuwen Yang, Zhihao Yang, Dong Li, Yingxueff Zhang, Zhanguang Zhang, Guojie Song, and Jianye Hao. Versatile multi-stage graph neural network for circuit representation. *Advances in Neural Information Processing Systems*, 35:20313–20324, 2022. 4

[37] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 2017. 1, 2