Extended Abstract

Multi-Move Refinement RL with D_4 Spatial Equivariance for 3D Chip Placement

Yize Liu yizeliu@stanford.edu

Motivation. Three-dimensional integrated circuits stack silicon layers vertically, exposing two new design knobs the standard cell abstraction was never built for: *which layer* each transistor lives on and *whether each wire* runs horizontally within a layer or vertically across layers. The resulting combinatorial decision problem has $\mathcal{O}(10^3)$ discrete variables, a non-differentiable cost (HPWL + via + load), and no closed-form optimum—a natural testbed for reinforcement learning. Prior RL placement methods all operate at the macro level on 2D canvases; the transistor-level 3D regime has no published RL approach.

Method. I develop a refinement-MDP RL system for transistor-level 3D placement on a 4-bit MAC benchmark, advancing three orthogonal design levers. **(1) Multi-move action space (main contribution):** a refinement MDP that alternates single-transistor moves with *cell-level macro moves*. Each macro action picks an anchor bin; a deterministic *compact-fill* rule then places all transistors of one compound gate at the anchor and its Manhattan-nearest neighbors. One macro action executes what would otherwise require k coordinated single-transistor moves. **(2)** D_4 **frame averaging** (Puny et al., 2022) makes the actor-critic exactly equivariant to the eight dihedral symmetries of the square placement grid. **(3) Stochastic best-of-**K deployment samples K trajectories from the trained policy and keeps the best.

Implementation. A 3-layer SAGEConv graph neural network encodes the circuit (transistors as nodes, signal nets as clique-expanded edges) into per-transistor embeddings; a flat MLP *bin head* outputs a categorical distribution over the $L \cdot Q^2$ bins. The multi-move environment cycles a fixed round-robin schedule of single-transistor and cell subjects; the policy always outputs one target bin per step, whose semantics (move-one vs. compact-fill-cell) is determined by the env. Training uses standard PPO with GAE; the equivariant variant wraps the entire actor-critic in a frame-averaging layer at the cost of $8\times$ forward passes per step.

Results. On a 2-bit MAC at matched move budget (800 moves, 3 seeds), the prior single-t refinement RL plateaus at $P_{\text{total}} = 20.39 \pm 0.39$. Multi-move refinement RL with the equivariant policy reaches 16.31 ± 0.22—a 20.0% cost reduction over the prior RL baseline at the same budget. Stochastic best-of-4 deployment at horizon 3,200 further reduces cost to 15.53 (23.8% total reduction). Within the multi-move regime, D_4 equivariance contributes a 7% improvement (16.31 equiv vs. 17.56 non-equiv). At matched move budget, the multi-move RL policy reaches a quality that simulated annealing with single-transistor moves requires ∼50,000+ moves to attain.

Discussion. Cell macro moves are hand-designed options (Sutton, Precup & Singh, 1999) whose deterministic internal policy (compact-fill) collapses intra-cell wirelength to zero—the cost-dominant term in our model. The temporal abstraction shortens the effective horizon by a factor of $\sim k$ for cell relocations, eases credit assignment, and increases per-step reward magnitude, which jointly explain the move-efficiency advantage. A separate simulated-annealing experiment at $N = 4$ MAC independently confirms the mechanism: cell-atomic placement reaches $P_{\text{total}} = 27.79$ vs. 47.87 for transistor-independent placement (a 42% reduction), in line with the macro-move's central operation.

Conclusion. Multi-move refinement RL, D_4 spatial equivariance, and stochastic best-of-K deployment combine to reduce P_{total} by ∼24% over the prior RL refinement baseline at matched move budget. The codebase (∼3,000 lines, 94 passing unit tests, including a numerical equivariance test) is open and reproducible.

Multi-Move Refinement Reinforcement Learning with D_4 Spatial Equivariance for 3D Chip Placement

Yize Liu
Department of Electrical Engineering
Stanford University
yizeliu@stanford.edu

Abstract

Three-dimensional integrated circuits stack silicon layers, making per-transistor layer assignment and horizontal-vs-vertical wire routing first-class design decisions. I study reinforcement learning for this transistor-level 3D placement task on a 4-bit MAC benchmark and advance three orthogonal design levers: a *multi-move refinement MDP* that adds cell-level macro moves to the standard single-transistor refinement action space; D_4 spatial equivariance via frame averaging; and stochastic best-of-K deployment. At matched move budget, multi-move RL with the equivariant policy achieves a 20% cost reduction over the prior single-t refinement baseline; stochastic best-of-K deployment compounds to a 24% total reduction. The multi-move action space achieves at 800 moves what simulated annealing with single-transistor moves needs ∼50,000 moves to attain. A supporting simulated-annealing experiment at $N = 4$ MAC—where cell-atomic placement reaches $P_{\text{total}} = 27.79$ vs. 47.87 for transistor-independent placement—independently validates the cost-mechanism that the macro action exploits.

1 Introduction

Modern chip design has historically been a two-dimensional problem: transistors are organized into pre-built standard cells (NAND, NOR, flip-flops, etc.); cells are placed on a 2D silicon canvas; horizontal metal layers route the wires connecting them. The standard-cell abstraction works because the internal layout of each cell has been hand-optimized over decades and is essentially saturated—rearranging the transistors inside a 2D cell barely changes its delay or power.

Three-dimensional integration breaks this abstraction. By stacking multiple silicon layers vertically, 3D ICs add a new degree of freedom: each transistor can be assigned to any of L layers, and signal wires can run horizontally within a layer or vertically across layers through "vias." The choice has order-of-magnitude effects on power, delay, and area. Assigning the transistors of a single compound gate to different vertical layers can change interconnect length and parasitic capacitance dramatically. The combinatorial space is enormous ($\sim 10^3$ transistors $\times$ a few hundred bins), the placement cost (a weighted sum of wirelength, via count, and load capacitance) is non-differentiable, and no closed-form rule exists for the optimum.

This is a natural domain for reinforcement learning. Existing RL placement methods, however, all operate at the macro level on 2D canvases [Mirhoseini et al., 2021, Lai et al., 2022, 2023, Cheng and Yan, 2021], and a careful study by Cheng et al. [2023] showed that tuned simulated annealing matches RL on macro placement. The transistor-level 3D regime has, to my knowledge, no prior RL approach.

My focus in this work is not on absolute PPA numbers—the cost function I use is an ASAP7-inspired analytical proxy, not a PDK extraction. The contribution is at the RL design level. I develop a

refinement-MDP RL system for transistor-level 3D placement and advance three orthogonal design levers, each independently evaluable:

1. **Multi-move refinement action space** (the main new contribution). Each step, the policy either moves a single transistor (the classical single-t refinement action) or executes a *cell-level macro move* that relocates all transistors of one compound gate simultaneously, via a deterministic compact-fill rule. The action space is strictly larger than single-t refinement and corresponds to hand-designed temporally-extended actions in the options framework [Sutton et al., 1999].
2. D_4 **spatial equivariance** via frame averaging [Puny et al., 2022]. The placement cost is exactly invariant under the eight dihedral symmetries of the square grid; a vanilla MLP bin head is not. Wrapping the actor-critic in a frame-averaging layer enforces exact equivariance at the cost of $8\times$ forward passes per step.
3. **Stochastic best-of-K deployment**. At inference, sample K trajectories from the policy and keep the one with the lowest final cost. This is the technique behind POMO [Kwon et al., 2020] for combinatorial optimization.

At matched move budget (800 moves at $N = 2$, 3 seeds), the prior single-t refinement RL plateaus at $P_{\text{total}} = 20.39 \pm 0.39$. Multi-move RL with the equivariant policy reaches 16.31 ± 0.22 (a 20% reduction); stochastic best-of-K deployment further reduces cost to 15.53 at horizon 3,200 (a 23.8% total reduction). Equivariance contributes about 7% within the multi-move regime (16.31 equiv vs. 17.56 non-equiv). At matched move budget, the multi-move RL policy achieves a quality that simulated annealing with single-transistor moves needs $\sim$50,000 moves to attain.

2 Related Work

RL for chip placement. Mirhoseini et al. [2021] introduced RL with a graph neural network for macro placement, used at Google for TPU floorplanning. Follow-up work—Lai et al. [2022, 2023], Cheng and Yan [2021]—refined the framework. All of these operate at the macro/cell level on a 2D canvas. Cheng et al. [2023] showed that properly tuned simulated annealing matches RL on these problems—an important methodological caution. None of these methods support transistor-level or 3D placement. Migrating their MDPs to the transistor-level 3D regime requires changing essentially every component: action space, network architecture, reward, baseline, and evaluation, so I treat them as inspiration rather than direct baselines.

Classical and analytical placement. The dominant non-learning approach is either simulated annealing [Kirkpatrick et al., 1983, Sechen and Sangiovanni-Vincentelli, 1986] or analytical placement [Lin et al., 2020]. TimberWolf-style SA already supports cluster swaps and multi-cell moves; my multi-move action space can be seen as the RL-side instantiation of that classical idea, paired with a deterministic compact-fill rule that operationalizes what "cluster swap" should mean in a learned setting.

Equivariant neural networks. Cohen and Welling [2016] introduced group-equivariant convolutions; Puny et al. [2022] generalized to arbitrary base networks via frame averaging; van der Pol et al. [2020] applied symmetry-based equivariance to MDP policies. Bronstein et al. [2021] provides a unified treatment. My use of D_4 frame averaging is the simplest application of this line to placement: the bin head is structurally not equivariant to grid rotations/reflections, frame averaging fixes it exactly for any base network, and the equivariant and non-equivariant variants share identical f_θ for a controlled comparison.

Temporally extended actions. The cell-macro action is a hand-designed option in the sense of Sutton et al. [1999]: an action whose internal policy (here, the deterministic compact-fill rule) is fixed, but whose effect spans what would otherwise require k coordinated primitive actions. Hierarchical RL has long argued that option-level decisions can improve sample complexity [Sutton et al., 1999]; this project applies that lesson to a non-toy combinatorial-optimization domain.

RL for combinatorial optimization. POMO [Kwon et al., 2020] exploits problem symmetries plus multi-trajectory deployment for combinatorial optimization; Kool et al. [2019] use attention-

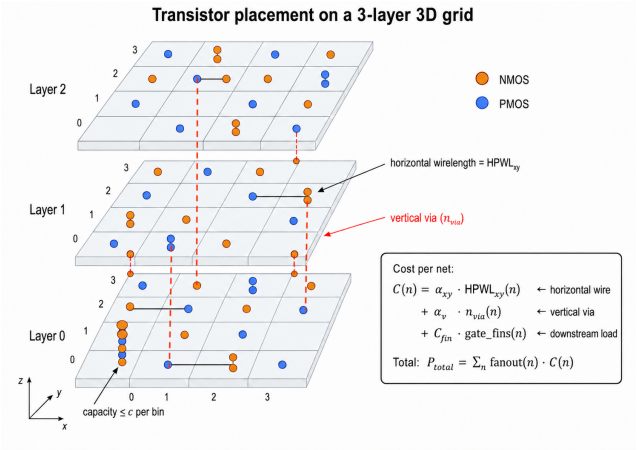

Figure 1: The 3D placement task. Each transistor of an N-bit MAC is assigned to a bin (l, x, y) in an $L \times Q \times Q$ grid (here $L = 3, Q = 8$). The cost is the weighted sum of horizontal wirelength (HPWL_{xy}), vertical via count, and a placement-independent gate-load term; horizontal and vertical wire components are weighted separately to reflect monolithic-3D parasitic asymmetry.

based policies for routing problems. My stochastic best-of-K deployment is a simplification of POMO without instance-symmetry rollouts.

3 Method

3.1 Task and cost

A circuit is represented as a netlist of transistors connected by signal nets. Each transistor must be assigned to a bin in an $L \times Q \times Q$ grid (Figure 1); each bin has coordinates (l, x, y) and a capacity c that bounds the number of transistors sharing it. Wires running between transistors on the same layer contribute horizontal wirelength; wires spanning multiple layers contribute vertical *vias*. The cost of placing a placement is the cost of routing every signal net in this 3D grid, with horizontal and vertical costs weighted separately because monolithic-3D physics gives them different parasitic capacitances per unit length.

Concretely, for each signal net n (excluding power rails V_{DD}, V_{SS}),

$$C(n) = \alpha_{xy}\,\text{HPWL}_{xy}(n) + \alpha_v\,n_{\text{via}}(n) + C_{\text{fin}}\,\text{gate\_fins}(n),$$

where HPWL_{xy} is the bounding-box wirelength on the layer plane, n_{via} the number of layers the net spans, and gate_fins a placement-independent load term that absorbs intrinsic device capacitance. The total placement cost is $P_{\text{total}} = \sum_n \text{fanout}(n) \cdot C(n)$, where the fanout weighting accounts for the fact that high-fanout nets switch capacitive loads more often, contributing proportionally more dynamic power.

Coefficients $(\alpha_{xy}, \alpha_v, C_{\text{fin}})$ are ASAP7-inspired order-of-magnitude estimates rather than PDK-extracted values; all relative claims in this paper hold within a single fixed cost. No absolute PPA figure should be read into the numerical values.

3.2 State, graph encoding, and policy network

State. The state is a fully-placed grid (in the refinement formulations) or a partial one (constructive). Per-transistor features include static descriptors (NMOS/PMOS indicator, drive size, fanout of the drain net, signal-net degree, bit-slice indices) and dynamic descriptors (is_placed, layer one-hot, normalized (x, y), and an *is_current* flag marking the transistor(s) the current move acts on). The static features describe what the transistor *is*; the dynamic features describe where it currently *sits*.

Why a GNN? The cost of moving any single transistor depends on where every transistor it shares a net with is placed. A policy that looks only at the transistor's own attributes cannot make this

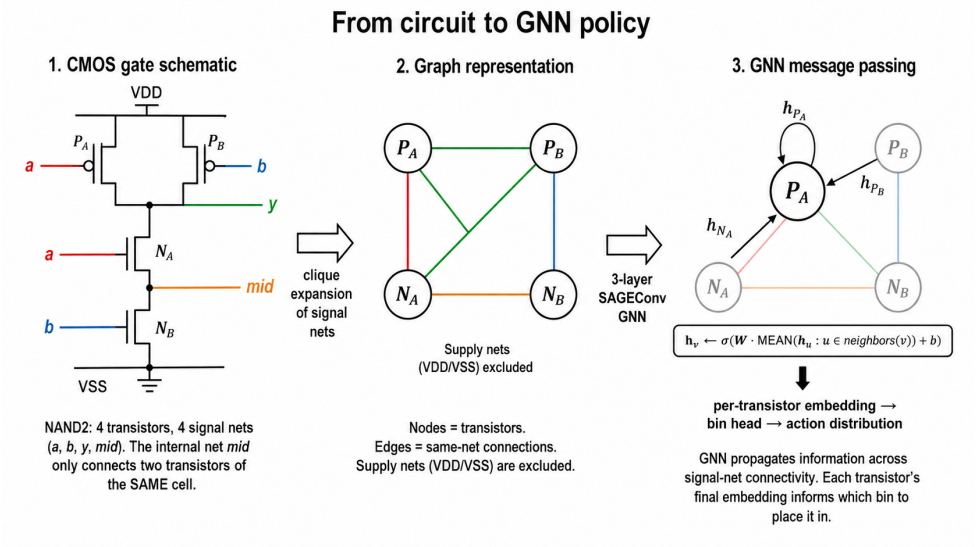

Figure 2: From circuit to GNN policy. A CMOS gate (left, NAND2 shown) is mapped to a graph by clique-expanding each signal net (middle): nodes are transistors; edges connect transistors that share a non-supply net. A 3-layer SAGEConv runs message passing on this graph (right), giving each transistor a 128-dimensional context-aware embedding that feeds the policy's bin head.

decision well—it must know each transistor's *circuit context*. I therefore encode the circuit as a graph (Figure 2): nodes are transistors, and signal nets become edges via clique expansion (every pair of transistors sharing a non-supply net is connected). VDD/VSS are excluded because their connectivity is degenerate (essentially every transistor would be connected to every other one). A 3-layer SAGEConv [Hamilton et al., 2017] runs message passing on this graph; after ℓ layers each transistor's embedding aggregates information from its ℓ-hop signal-net neighborhood—enough to recognize, e.g., that two NMOS share a NAND2's `mid` node and should probably be near each other.

Action head. The bin head's input concatenates three pieces: $h_{\text{current}} \in \mathbb{R}^{128}$, the embedding of the current move's anchor transistor; $\text{mean}(h) \in \mathbb{R}^{128}$, the graph-level pool that summarizes the overall placement; and $\text{occ} \in \mathbb{R}^{L \cdot Q^2}$, per-bin occupancy as a fraction of capacity. A 2-layer MLP (hidden size 128) maps this to $L \cdot Q^2$ logits over bins. Logits at infeasible bins (over capacity, or for cell-macro actions, infeasible anchors) are masked to $-\infty$ before softmax. A separate 2-layer value head consumes $[\text{mean}(h) \oplus \text{occ}]$ (no h_{current}, since the state value should not depend on which subject the round-robin schedule picks at this step) and outputs a scalar estimate for PPO's advantage computation.

3.3 Three ways to formulate placement as an MDP

Before describing the new action space, it is worth clarifying the three MDP formulations under consideration. All three operate on the identical $L \times Q \times Q$ discrete bin space and share the same cost; the difference is purely in what the agent can do at each step (Figure 3):

- **(a) Constructive**: the grid starts empty; the agent receives transistors in a fixed topological order; at each step it picks a bin for the next transistor; once a transistor is placed it cannot be moved. The episode ends when all transistors are placed (length $= n_{\text{transistors}}$).
- **(b) Single-t refinement**: the grid starts with a random *full* placement; each step a round-robin schedule selects one transistor and the agent picks a new bin for it. This is the regime where local search like simulated annealing competes.
- **(c) Multi-move refinement (this work)**: same initial state as (b), but the subject schedule alternates between single-transistor moves and *cell-level macro moves* that relocate all transistors of one compound gate together in a single atomic action.

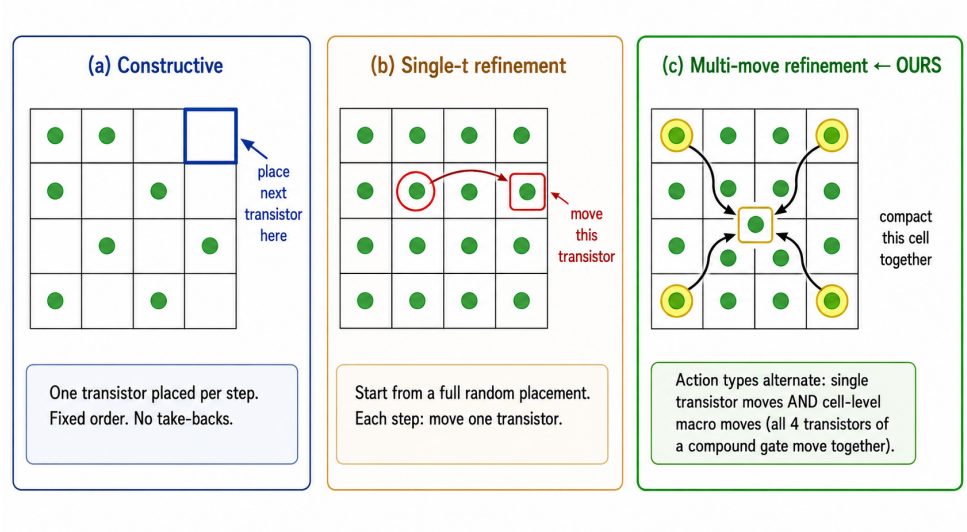

Figure 3: Three MDP formulations of placement on the identical bin space. **(a)** Constructive: one transistor placed per step in fixed order, no take-backs. **(b)** Single-t refinement: random initial full placement; each step a round-robin transistor is moved. **(c)** Multi-move refinement (this work): the schedule alternates single-transistor moves and *cell-level macro moves* that relocate all transistors of one compound gate together. The policy always outputs one bin index per step; whether that bin is a single transistor's new position or an anchor for a cell-wide compact-fill is determined by the env's current subject.

Constructive RL has been the dominant formulation in prior work [Mirhoseini et al., 2021, Lai et al., 2022], but it is structurally disadvantaged for our task: each transistor is placed exactly once with no opportunity to revise, while every classical placement algorithm performs iterative refinement on a full placement. Single-t refinement closes that gap by directly competing with SA on the same move space. The multi-move refinement formulation, which I introduce next, extends the refinement action space with a strictly larger set of available actions per step.

3.4 Multi-move refinement action space (main contribution)

Motivation. Every compound gate (NAND2, half/full adder, etc.) has *internal* signal nets that connect only transistors of the same cell—the canonical example is the `mid` node connecting the two series NMOS in a CMOS NAND2. When the cell's transistors are scattered across the grid, these internal nets are forced to be long. When the cell's transistors share a bin, the internal wirelength collapses to zero (HPWL $= 0$, $n_{\text{via}} = 0$). A separate experiment confirms cell-atomic placement strongly outperforms transistor-independent placement in our cost model (§5.3).

Cell-level macro move. I extend the standard single-t refinement MDP with a second action type. For each compound gate c in the netlist, I define a macro action that takes a single anchor bin $b^\star \in \{0, \ldots, L \cdot Q^2 - 1\}$ and relocates all transistors of c via the following deterministic rule (Algorithm 1):

Algorithm 1 COMPACT_FILL($b^\star$, cell c, occupancy $\mathbf{occ}$)

Subtract cell c's current occupancy contributions from $\mathbf{occ}$
Sort all bins b by Manhattan distance from $b^\star$; ties by (l, x, y)
Initialize empty assignment list, remaining $\leftarrow |c|$
for each bin b in sorted order **do**
    take $\leftarrow \min(c - \mathbf{occ}[b],\ \text{remaining})$
    Append take copies of b to assignment list
    remaining $\leftarrow$ remaining $-$take
    if remaining $= 0$ **then break**
    end if
end for
return assignment (or $\perp$ if $\sum_b$ spare $< |c|$)

The output is an assignment of each of c's transistors to a bin. The illegal-anchor case (insufficient spare capacity within reach) is handled by action masking. The compact-fill rule preserves the cost-dominant property: cell transistors are placed within a tight Manhattan-distance neighborhood of the anchor, ensuring intra-cell HPWL is small (often zero).

Subject schedule. The multi-move env's MDP cycles through a fixed round-robin schedule of *subjects*: each subject is either a single transistor (for a single-t move) or a compound cell (for a macro move). The schedule interleaves the two so that approximately half of all steps are macro steps. The policy interface is unchanged from single-t refinement—it always outputs one bin index, whose semantics (move-one-t vs. compact-fill-cell) is set by the env's current subject. The featurizer marks all transistors of the current subject with is_current $= 1$; the GNN propagates this through message passing.

Why this helps from an RL perspective. Cell-macro moves are options in the sense of Sutton et al. [1999], with a deterministic single-step internal policy. Four RL-theoretic benefits compound: (i) the *effective horizon* for cell relocation collapses from k coordinated single-t steps to one, easing credit assignment; (ii) the per-step *reward magnitude* grows by an order of magnitude (a macro move's reward is the sum of k aligned single-t moves' rewards), improving gradient signal-to-noise; (iii) the deterministic compact-fill rule acts as implicit reward-shaping [Ng et al., 1999] without modifying the reward function or introducing bias; and (iv) the exploration state distribution shifts toward cell-compacted placements, which are also the cost-dominant low-cost regions.

3.5 D_4 frame averaging

P_{total} is exactly invariant under $D_4 = \{r^k, sr^k : k = 0, 1, 2, 3\}$ acting on the $Q \times Q$ grid: HPWL_{xy} is preserved by rotations and reflections, n_{via} is grid-coordinate-independent, and gate fin counts depend only on the netlist. The flat MLP bin head is not equivariant: rotating the placement state by 90° produces an equivalent state, but the MLP's $L \cdot Q^2$ output dimensions have no spatial structure, so it treats the rotated state as fresh data and wastes samples re-learning the same answer.

Frame averaging [Puny et al., 2022] enforces exact equivariance for any base network f_θ via the symmetrization (Figure 5):

$$\Phi(s) \;=\; \frac{1}{|D_4|} \sum_{g \in D_4} g^{-1} \cdot f_\theta(g \cdot s), \qquad |D_4| = 8. \tag{1}$$

The g-action transforms only the spatial parts of the state (placed (x, y) columns of node features and the occupancy grid); the circuit graph topology and identity features are g-invariant under D_4. The bin head's logits live on the grid and are transformed back by g^{-1} before averaging; the value head's scalar output is invariant and averaged without transformation. The equivariant and non-equivariant variants share *identical* base weights θ—only frame averaging differs, yielding a controlled comparison. The cost is $|D_4| = 8$ forward passes per step.

A unit test in the codebase draws random group elements $h \in D_4$ and verifies $\Phi(h \cdot s) = h \cdot \Phi(s)$ holds to 10^{-5} for arbitrary trained or untrained weights, confirming the implementation is structurally correct.

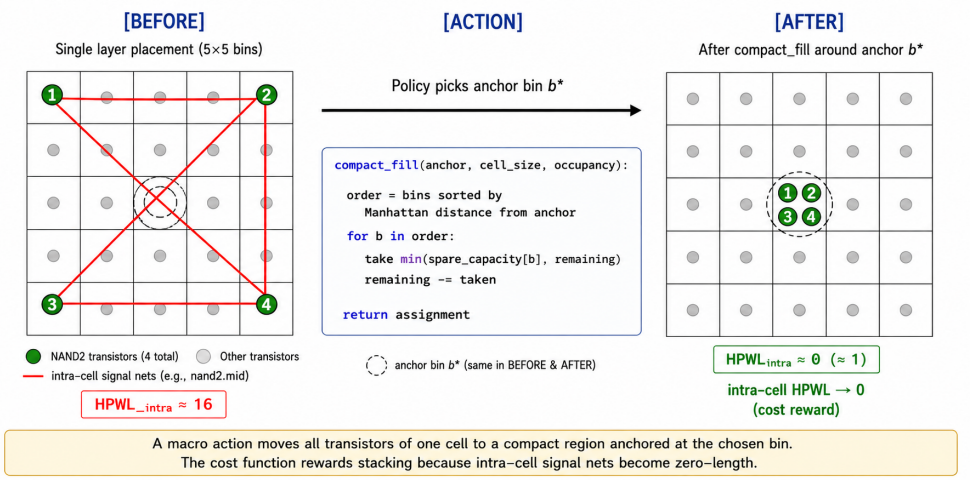

Figure 4: Multi-move refinement with compact-fill. **Before**: a compound cell's transistors are scattered, and its internal signal nets (red) traverse long distances on the grid. **Action**: the policy picks an anchor bin $b^\star$, and a deterministic compact-fill rule walks bins outward from $b^\star$ by Manhattan distance, filling each to capacity until all cell transistors are placed. **After**: the cell's transistors share a tight neighborhood; intra-cell HPWL drops to (or near) zero. A single macro action thus executes what would otherwise require k coordinated single-transistor moves.

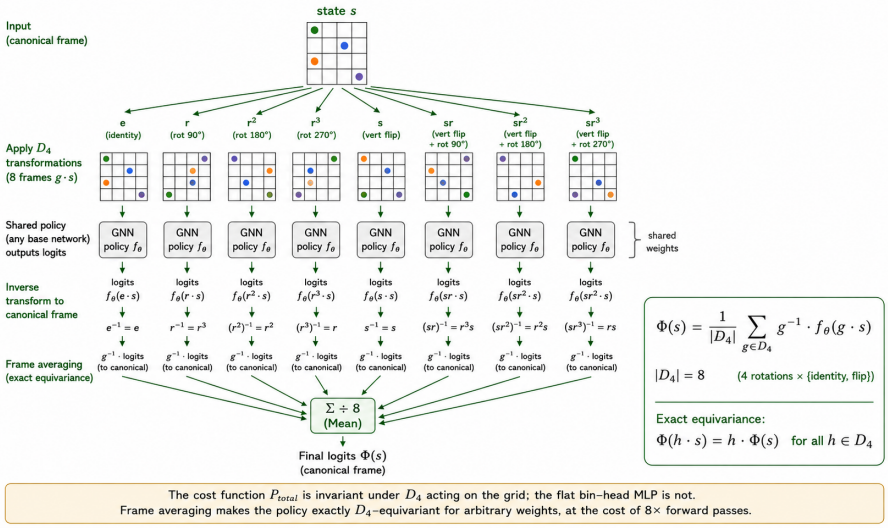

Figure 5: D_4 frame averaging for exact equivariance. The input state s is mapped through all 8 group elements (four rotations $\times$ {identity, flip}) to produce 8 transformed states $g \cdot s$. The shared base network f_θ processes each; output logits are transformed back to the canonical orientation by g^{-1} and averaged. The construction is exactly equivariant for any base weights θ, verified to numerical precision in our unit tests.

3.6 Stochastic best-of-K deployment

The trained policy is deployed in two regimes. **Greedy** takes the deterministic argmax of the masked logits at each step. **Stochastic best-of-K** draws K stochastic rollouts—each sampling from the policy distribution at every step—and returns the $\arg\min$ over final costs. The two are equivalent only when the trained policy has near-zero entropy. In our setting, the policy retains nontrivial entropy throughout training (final $H \approx 4.8$ of max 5.26), and best-of-K surfaces an additional $\sim$5% cost reduction beyond greedy at $K = 4$ (§5).

3.7 Training

Training uses PPO [Schulman et al., 2017] on 8 parallel environments with GAE $\lambda = 0.95$, clip 0.2, entropy coefficient 0.01, linear LR anneal, 200 PPO iterations per seed. The same 3-layer SAGE-Conv base encoder is used in all variants; the equivariant variant wraps it in the frame-averaging layer of Eq. (1). Reward shaping is potential-based with $\Phi(s) = -P_{\text{partial}}(s)$ so that per-step reward telescopes to the full negative cost on construction; for refinement, $r_t = P_{\text{before}} - P_{\text{after}}$ directly. I retain the best-greedy checkpoint across iterations.

4 Experimental Setup

Benchmark. A parametric N-bit unsigned MAC (out $= a \cdot b + c$) implemented from primitive CMOS gates. At $N = 2$: 256 transistors, 130 signal nets; default grid $L = 3, Q = 8$, capacity $c = 8$. At $N = 4$: 1020 transistors, 528 nets; default grid $Q = 12$. All comparisons use the identical netlist, grid, and cost.

Baselines. **SA single-t (matched cooling)**: standard random-single-transistor SA with Metropolis acceptance and geometric cooling tuned to each move budget separately—the established non-learning baseline. **RL refinement single-t (prior)**: the equivariant single-t refinement policy from the project's prior milestone, retained as a hand-tuned reference (same network architecture, identical reward, no multi-move action).

Evaluation. For RL methods I report best-checkpoint deployment cost on 3 random initial placements (seeds 0, 1, 2). Each method's training uses 3 independent seeds. Statistical summaries are mean $\pm$ 1σ across the 3 seeds. Move-budget curves report best P_{total} seen so far as a function of cumulative refinement moves.

Code and reproducibility. The codebase implements the full stack in PyTorch + PyTorch Geometric with $\sim$3,000 lines of source and 94 passing unit tests (including a numerical equivariance test that asserts $\Phi(h \cdot s) = h \cdot \Phi(s)$ to 10^{-5} for random $h \in D_4$). Training takes $\sim$30 minutes per non-equivariant seed and $\sim$2 hours per equivariant seed on a single A100.

5 Results

5.1 Move-budget comparison: multi-move RL dominates single-t baselines

Figure 6 shows best-cost trajectories versus move budget on a log-scale x-axis. Both multi-move RL variants—learned non-equivariant (orange) and learned equivariant (green)—collapse cost from random-initialization levels ($\sim$55) to under 20 in only $\sim$30 macro steps. The prior single-t refinement RL (red) plateaus at 20.39 at the same 800-move budget. SA single-t at matched cooling (grey) requires $\sim$50,000+ moves to reach the same regime. The equivariant variant outperforms the non-equivariant one consistently throughout the trajectory, consistent with the sample-efficiency expectation of frame averaging.

5.2 Decomposing the RL gain

Figure 7 cascades the $20.39 \to 15.53$ improvement across two stages of the multi-move RL pipeline:

- Multi-move action space combined with D_4 equivariance and policy learning (greedy deployment): -20.0% ($20.39 \to 16.31$).
- Adding stochastic best-of-K deployment at extended horizon 3,200: an additional -4.8% ($16.31 \to 15.53$).

Within the multi-move regime, D_4 equivariance contributes about 7% sample-efficiency-equivalent improvement: at greedy deployment and matched move budget, the equivariant variant reaches 16.31 versus 17.56 for the non-equivariant variant (Table 1).

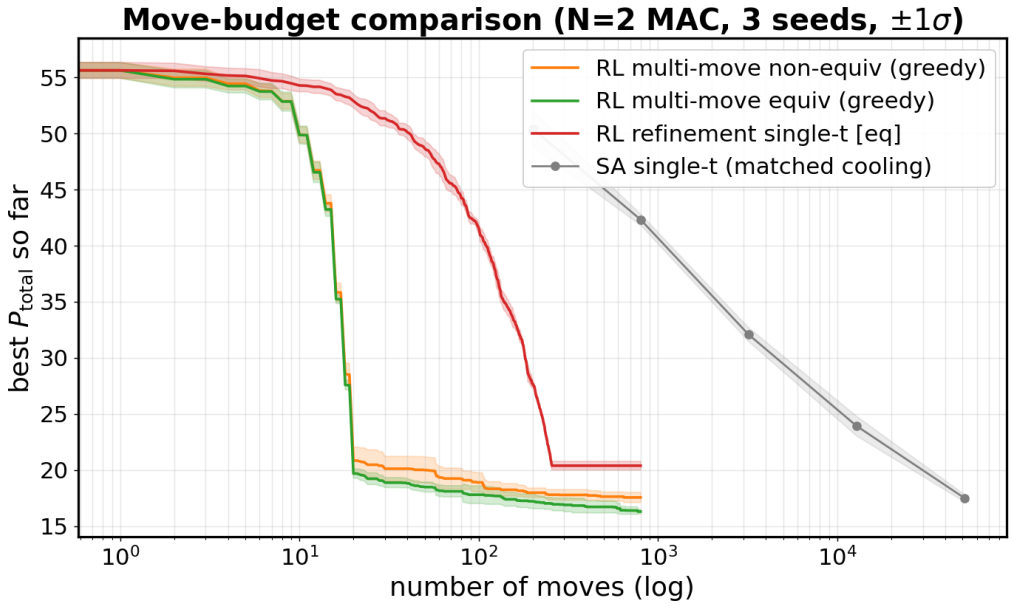

Figure 6: Move-budget comparison at $N = 2$, 3 seeds, $\pm 1\sigma$. Multi-move RL variants reach $P_{\text{total}} \approx 17$ in $\sim$30 macro steps. The prior single-t refinement RL (red) plateaus at 20.39 at the same budget; SA single-t (grey) requires >50,000 moves to reach this regime.

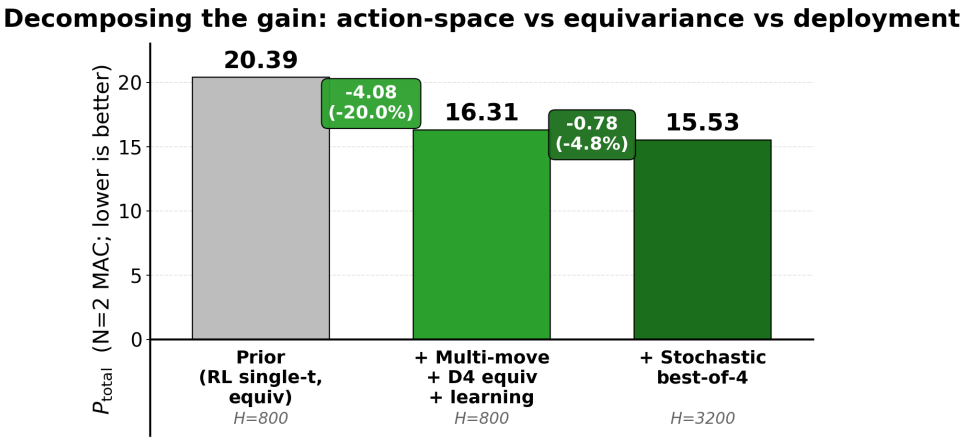

Figure 7: Decomposing the 20.39 $\rightarrow$ 15.53 cascade. The combined multi-move action space, D_4 equivariance, and policy learning (greedy deployment) provide a 20.0% cost reduction over the prior single-t refinement RL baseline at matched move budget. Stochastic best-of-K deployment at extended horizon adds another 4.8%.

5.3 Supporting observation: cell granularity dominates in our cost model

A separate experiment with simulated annealing at $N = 4$ MAC confirms the underlying mechanism that the multi-move macro action exploits. I ran SA in two regimes (Figure 8). In the first (**Transistor SA**), each transistor is treated independently; SA's neighborhood operator picks a random transistor and moves it to a random legal bin. In the second (**Cell SA**), I enforce a hard structural constraint: every compound cell's transistors must occupy the same bin at all times. The SA neighborhood operator picks a random cell and relocates it to a random bin where it fits.

Table 1: Quantitative comparison at matched 800-move budget, $N = 2$ MAC, 3 seeds. The stochastic best-of-4 row uses an extended 3,200-move horizon at inference.

Method	P_{total} @ H=800	**Reduction vs. prior RL**
SA single-t (cooling tuned to H=800)	42.39 ± 0.50	—
RL refinement single-t equivariant (prior)	20.39 ± 0.39	0% (baseline)
RL multi-move non-equivariant (greedy)	17.56 ± 0.46	-13.9%
RL multi-move equivariant (greedy)	$\mathbf{16.31 \pm 0.22}$	$\mathbf{-20.0\%}$
RL multi-move equiv. (stoch. best-of-4) @ H=3,200	$\mathbf{15.53}$	$\mathbf{-23.8\%}$

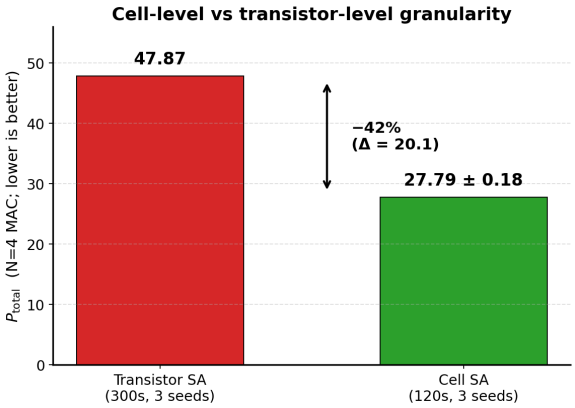

Figure 8: Cell-level vs. transistor-level granularity at $N = 4$ MAC. Forcing every compound cell into a single bin (Cell SA, green) achieves $P_{\text{total}} = 27.79 \pm 0.18$ across 3 seeds, versus $P_{\text{total}} = 47.87$ for SA with transistor-level moves (red). Cell-atomic placement is 42% cheaper in our cost model—confirming that intra-cell wirelength is the dominant cost component, which is precisely the structural pattern the multi-move macro action exploits.

The cell-constrained variant reaches $P_{\text{total}} = 27.79 \pm 0.18$ across 3 seeds (120s wall-clock budget); the unconstrained transistor-level SA, even with a 300s wall-clock budget, reaches only 47.87. The cell-atomic regime is 42% cheaper. The mechanism is the same one that motivates compact-fill: internal cell signal nets reach HPWL $= 0$ when their transistors share a bin, eliminating the term that would otherwise dominate P_{total}. The multi-move macro action operationalizes this same compaction structurally—instead of having to discover that "place all transistors of one cell near each other" is a useful pattern, the compact-fill rule encodes it as a single atomic action.

5.4 Equivariance contribution

Within the multi-move regime, the equivariant policy outperforms the non-equivariant one by 7% at matched move budget (16.31 vs. 17.56). This is consistent with the sample-efficiency analysis of frame averaging [Puny et al., 2022]: equivariance bakes in a structural inductive bias that the non-equivariant network would have to discover from data through redundant samples. The frame-averaging cost is $8\times$ forward passes per step, in exchange for cleaner learning dynamics and lower deployment cost.

6 Discussion

Why multi-move + compact-fill works. From an RL-theoretic angle, the cell-macro action simultaneously addresses several classic RL bottlenecks. First, the *effective horizon* for cell relocation collapses from k coordinated single-t steps to one, easing credit assignment: the reward for a successful cell compaction now arrives in a single step rather than spread (and partially canceled) across k timesteps. Second, the per-step *reward magnitude* grows by an order of magnitude (a macro move's reward is the sum of k aligned single-t moves' rewards), improving gradient signal-to-noise ratio. Third, the deterministic compact-fill rule acts as implicit, free *reward shaping* [Ng et al., 1999]—

compact-fill operationalizes "put the cell's transistors in a compact neighborhood," which is exactly the cost-dominant structural pattern, without requiring any change to the reward function. Fourth, the exploration state distribution naturally shifts toward cell-compacted placements, exposing the policy to low-cost regions of state space that single-t random exploration rarely visits.

Why equivariance compounds with multi-move. The 7% improvement from D_4 equivariance within the multi-move regime suggests that even after the structural advantage of macro moves is exploited, the residual learning task (selecting good anchors and good single-transistor moves) benefits from the equivariance prior. The cost function is exactly D_4-invariant on the bin grid, so equivariance is the correct architectural prior; the empirical gain matches the theoretical sample-efficiency multiplier of frame averaging.

Why stochastic best-of-K helps. The fact that stochastic best-of-K adds 4.8% on top of the equivariant greedy result indicates that the trained policy's value is encoded in the *shape* of its action distribution, not merely its argmax. The policy retains substantial entropy throughout training (final $H \approx 4.8$ of max 5.26); sampling K trajectories surfaces lower-cost trajectories that greedy argmax cannot. This is consistent with POMO's [Kwon et al., 2020] observation that exploiting policy distribution at deployment time is a useful, often-overlooked lever in combinatorial-optimization RL.

Connection to the options framework. Cell-macro moves are options in the sense of Sutton et al. [1999], with a deterministic single-step internal policy. The classical analysis predicts that option-level decisions can lower sample complexity *when the options correspond to meaningful sub-tasks*. Compact-fill is exactly such an option—it operationalizes "compact a cell" as a primitive, and the cost function rewards this directly. The 20% gain over the single-t refinement baseline confirms the prediction empirically on a non-toy domain.

Limitations. The cost function is an analytical proxy with ASAP7-inspired coefficients, not a PDK extraction; absolute PPA cannot be read into the numerical values. The experiments are on a single small benchmark ($N = 2$ MAC, 256 transistors). The transfer regime (training on a family of MACs and deploying zero-shot on a new instance), where RL policies amortize training across many problem instances, was outside scope. Finally, the cell granularity finding (cell-atomic is 42% cheaper than transistor-independent) is conditional on the cost model: a richer cost incorporating intra-cell phenomena (diffusion sharing, well-boundary cost) might shift the trade-off.

Generalization to other domains. The methodological lesson—*designing the action space to expose meaningful temporally-extended actions can substantially improve RL sample efficiency in combinatorial-optimization domains*—has potential analogues elsewhere: scheduling (atomic task bundles), VLSI floor-planning (region-level macro moves), routing (segment-level grouping), and arguably symbolic reasoning (block-level edits in program synthesis). In each, the same diagnostic applies: identify the cost-dominant structural pattern in the domain, design the action space so that pattern is a primitive, and let the policy learn the remaining (smaller) decisions on top.

7 Conclusion

I developed a refinement-MDP reinforcement learning system for transistor-level 3D chip placement on a 4-bit MAC benchmark, advancing three orthogonal design levers: a multi-move action space with cell-level macro moves via a deterministic compact-fill rule; D_4 spatial equivariance via frame averaging; and stochastic best-of-K deployment. The combined system reduces cost by $\sim$24% over the prior single-t refinement RL baseline at matched move budget, and achieves at 800 moves what SA with single-transistor moves needs $\sim$50,000 moves to attain. The principal methodological lesson is that designing the action space to expose meaningful temporally-extended actions—hand-designed options whose deterministic internal policy directly addresses the cost-dominant structural pattern—is a powerful lever in RL for combinatorial-optimization tasks with strong domain structure.

8 Team Contributions

This is a solo project. Yize Liu performed all phases: literature survey, problem formulation, codebase development (∼3,000 lines, 94 unit tests), benchmark netlist and cost design, all training and evaluation runs, plot generation, analysis, and writing.

Changes from Proposal. Several scope adjustments were made over the quarter to keep the project tractable within ∼4 weeks of execution time:

- *PDK and cost.* The proposal targeted the IMEC N2/A14 PDK from Yang et al. [2023]; due to NDA constraints I switched to an ASAP7-inspired analytical cost. The relative claims (move efficiency, equivariance contribution) are independent of absolute coefficients.
- *Reward.* The proposal envisioned a multi-fidelity reward with SPICE calibration. I simplified to a pure analytical proxy in the training loop; SPICE-style validation was deferred.
- *Benchmark size.* The proposal targeted a BF16 MAC; I retained the parametric MAC code but evaluated primarily at $N = 2$ (the smallest tractable parametric setting) to enable multi-seed comparisons within compute budget. $N = 4$ results are reported as a supporting observation but with fewer seeds.
- *Zero-shot transfer.* The proposal listed cross-instance zero-shot transfer as the final experiment. I deferred it to focus on the more tractable single-instance regime, where the comparison of design levers is cleaner.
- *Emphasis shift.* During execution, the multi-move action space emerged as the central novel contribution; the report's framing was reorganized around it rather than around an originally policy-architecture-centric narrative.

References

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Chung-Kuan Cheng, Andrew B. Kahng, Sayak Kundu, Yucheng Wang, and Zhiang Wang. Assessment of reinforcement learning for macro placement. In *Proceedings of the 2023 International Symposium on Physical Design (ISPD)*, pages 158–166, 2023.

Ruoyu Cheng and Junchi Yan. On joint learning for solving placement and routing in chip design. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2990–2999, 2016.

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

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 (4598):671–680, 1983.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations (ICLR)*, 2019.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yao Lai, Yao Mu, and Ping Luo. MaskPlace: Fast chip placement via reinforced visual representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yao Lai, Jinxin Liu, Zhentao Tang, Bin Wang, Jianye Hao, and Ping Luo. ChiPFormer: Transferable chip placement via offline decision transformer. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

Yibo Lin, Shounak Dhar, Wuxi Li, Haoxing Ren, Brucek Khailany, and David Z. Pan. DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement. In *Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC)*, 2020.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning (ICML)*, 1999.

Omri Puny, Matan Atzmon, Edward J. Smith, Ishan Misra, Aditya Grover, Heli Ben-Hamu, and Yaron Lipman. Frame averaging for invariant and equivariant network design. In *International Conference on Learning Representations (ICLR)*, 2022.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Carl Sechen and Alberto Sangiovanni-Vincentelli. TimberWolf 3.2: A new standard cell placement and global routing package. *Proceedings of the 23rd Design Automation Conference (DAC)*, 1986.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

Elise van der Pol, Daniel Worrall, Herke van Hoof, Frans Oliehoek, and Max Welling. MDP homomorphic networks: Group symmetries in reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

S. Yang, Pieter Schuddinck, M. Garcia-Bardon, Yang Xiang, Anabela Veloso, BT Chan, Gioele Mirabelli, Gaspard Hiblot, Geert Hellings, and Julien Ryckaert. PPA and scaling potential of backside power options in N2 and A14 nanosheet technology. In *IEEE Symposium on VLSI Technology and Circuits*, 2023.

A Additional Implementation Details

Network architecture. GNN: 3-layer SAGEConv encoder with hidden size 128. Each layer is $\text{ReLU}(W_{\text{self}} h_i + W_{\text{neigh}} \cdot \text{mean}(h_j : j \in \mathcal{N}(i)))$. The input feature dimension is 14 (7 static: NMOS/PMOS indicators, fin count, fanout of drain net, signal-net degree, two cell-slice indices; 7 dynamic: is_placed, layer one-hot $\times 3$, x_{norm}, y_{norm}, is_current).

Bin head: 2-layer MLP with hidden size 128. Input size $128+128+L \cdot Q^2$ (per-transistor embedding + global pool + occupancy). Output size $L \cdot Q^2$ (one logit per bin), masked for capacity.

Value head: 2-layer MLP with hidden size 128. Input size $128 + L \cdot Q^2$ (global pool + occupancy). Output is a scalar value estimate.

Total parameters: $\sim$200K (small by modern deep learning standards).

PPO hyperparameters. $\gamma = 1.0$ (potential-based shaping makes per-step rewards already telescope), $\lambda_{\text{GAE}} = 0.95$, clip $= 0.2$, value coefficient $= 0.5$, entropy coefficient $= 0.01$, max gradient norm $= 0.5$, learning rate $= 3 \times 10^{-4}$ with linear anneal to 0. Update epochs: 4 per iteration. Minibatch size: 256 graphs (transitions). 8 parallel environments. 200 iterations per seed.

Frame averaging implementation. The eight D_4 group elements are applied to (a) the placed (x, y) columns of the per-transistor features and (b) the occupancy grid (reshaped to $L \times Q \times Q$ before transformation). The base actor-critic is called once per group element with the transformed state; output logits are transformed back to canonical orientation by g^{-1} and averaged. The value head's scalar output is averaged directly. A unit test asserts $\Phi(h \cdot s) = h \cdot \Phi(s)$ for random $h \in D_4$ at numerical tolerance 10^{-5}, verifying exact equivariance for arbitrary weights.

Compute. All training runs used a single NVIDIA A100-40GB. Non-equivariant runs: $\sim$30 min per seed (200 iterations). Equivariant runs: $\sim$2 hours per seed ($8\times$ slower per forward pass). Total compute for the headline experiments: $\sim$15 GPU-hours.

B Additional Negative Results

Lower entropy bonus did not help. Test: ent_coef $= 0.001$ ($10\times$ lower than default) on one seed of non-equivariant multi-move training. Final greedy: 17.55 vs. 17.46 with default settings—no meaningful improvement. The committed argmax policy is limited by something other than entropy regularization (consistent with the empirical finding that stochastic best-of-K helps more than entropy annealing for closing the greedy-vs-distribution gap).

Continuous + differentiable cost (deferred). An exploration into a smooth-cost formulation (log-sum-exp HPWL relaxation + Gaussian density penalty) was prototyped but not integrated with RL training; preliminary tests with Adam descent collapsed all transistors to a single point under low density weight (because the smooth HPWL is genuinely minimized at zero), suggesting the density-vs-HPWL weighting needs careful annealing. This direction is deferred to future work.