Journal of VLSI circuits and systems
RESEARCH ARTICLE
Journal of VLSI Circuits and Systems, ISSN: 2582-1458 Vol. 8, No. 1, 2026 (pp. 9-19)
WWW.VLSIJOURNAL.COM

Auto-PPA: An Adaptive Deep RL Agent for VLSI Physical Design Optimization

Hussain Ali Mutar[1], Ibtihal Razaq Niama ALRubeei[2], Omar Hashim Yahya[3], Naseer Ali Hussien[4], Haider TH. Salim AlRikabi[2*], Abdul Hadi M. Alaidi[1]

[1]Computer Department, College of Computer Science and Information Technology, Wasit University, Wasit, Iraq.
[2]Electrical Engineering Department, College of Engineering, Wasit University, Wasit, Iraq.
[3]Department of Computer Engineering Technology, Technical Engineering College for Computer and AI, Northern Technical University, Mosul, Iraq.
[4]Department of Computer Engineering Technology, College of Technical Engineering, Al-Ayen Iraqi University, AUIQ, Nasiriyah, Iraq.

KEYWORDS:
VLSI;
Optimization;
AI model;
Reinforcement learning;
Real-time PPA co-optimization

ARTICLE HISTORY:
Received: 17.11.2025
Revised: 12.12.2025
Accepted: 24.12.2025

DOI:
https://doi.org/10.31838/JCVS/08.01.02

Abstract

The physical design phase of Very-Large-Scale-Integration (VLSI) is notoriously difficult since it must strike a balance between PPA, power, and performance. Computationally costly design cycles and less-than-ideal Pareto fronts are common challenges of using traditional optimization methods to tackle these metrics in order. As part of physical design, this study suggests a new reinforcement learning (RL) framework that can optimize all three PPA measures in real time. In the proposed method, commercial electronic design automation (EDA) tools were used in conjunction with a deep deterministic policy gradient (DDPG) agent to make routing and placement decisions incrementally. Guided by a customized reward function that dynamically balances PPA trade-offs based on design stage priorities, the agent operates on a continuous action space that represents geometric coordinates and constraint modifications. While conventional sequential optimization methodologies reduce optimization runtime by about 35%, the proposed RL agent improves the power-performance product by 18.7% and the area reduction by 12.3%, according to simulation results on the ISPD 2015 benchmark suite. An innovative approach to optimize intelligent, adaptable physical designs that successfully traverse the high-dimensional PPA trade-off space is presented by the suggested framework.

Authors' e-mail IDs: hmutar@uowasit.edu.iq; Ibtihal.Razaq@uowasit.edu.iq; Omer_h_yahya@ntu.edu.iq; naseerali@alayen.edu.iq; hdhiyab@uowasit.edu.iq; alaidi@uowasit.edu.iq

Authors' ORCID IDs: 0009-0004-4934-416X; 0009-0003-8958-8487; 0000-0003-0277-562X; 0000-0001-9499-6694; 0000-0002-5201-2084; 0000-0001-9120-708X

How to cite this article: Hussain Ali Mutar et al. Auto-PPA: An Adaptive Deep RL Agent for VLSI Physical Design Optimization, Journal of VLSI Circuits and System, Vol. 8, No. 1, 2026 (pp. 9-19).

Introduction

The complexity of Very-Large-Scale-Scale Integration (VLSI) physical design has grown enormously due to unrelenting scaling of technology, which is now well into the sub-7 nm region.[1,2] At this stage, which is responsible for translating a logical circuit model into a physical architecture that can be manufactured, three goals—power consumption, timing performance, and silicon area utilization—must be optimized at the same time, despite their inherent conflicts.[3,4] As a result of using sequential or weighted-sum approaches, which do not adequately account for complex interdependencies and frequently converge to less-than-ideal solutions, traditional Electronic Design Automation (EDA) technologies are unable to solve this multidimensional optimization problem.[5,6] The advent of AI, and RL in particular, presents a game-changing chance to rethink physical design optimization as a learning-driven adaptive process that can find better trade-offs in this very limited design space.[7,8]

Literature Review

The use of machine learning to address different EDA problems has become increasingly popular in recent years. Supervised learning has been investigated in the

past for predictive modeling of optimization results and circuit parameters; Though evolutionary algorithms have been used for multiobjective optimization, they have low scalability.[9-11] Previous research in physical design has focused on optimization of placement for individual goals, such as wirelength minimization, and on-time closure using gradient-based approaches. However, these methods do not have the flexibility to adjust to new design-state circumstances; they usually just deal with PPA measurements or use set priority schemes. Table 1 provides a concise summary of the most relevant publications along with their limitations in relation to the suggested strategy. An examination of related works in the field of VLSI physical design optimization is shown here.

Limitations of Current Approaches

The existing physical design methods in industry use a combination of iterative refinement heuristics and constraint-driven algorithms, which have several serious flaws that worsen with technology. Missing chances for global co-optimization that could provide better Pareto fronts, these methods usually optimize timing and power first, with area being a mostly passive constraint.[14-16] Due to the significant computational overhead and longer design cycles caused by the iterative refinement process, which converges after numerous full tool runs (including placement, routing, timing analysis, and power analysis), the technique is not always efficient.[17,18] Also, because they cannot explore very far, gradient-based approaches and greedy algorithms sometimes converge to less-than-ideal local minima in the high-dimensional PPA landscape. Most importantly, these optimization algorithms are static and cannot change their focus based on the progress of the optimization process. For example, in optimization process, the initial focus might be on timing convergence, which later moves to power and area recovery.[11,18-20]

Proposed Solution and Contributions

While pioneering works such as that of Mirhoseini et al.[12,21] demonstrated RL for macro placement; they optimize primarily for proxy metrics such as the wirelength. On the other hand, Auto-PPA performs direct, simultaneous optimization of the final PPA signoff metrics (Power, WNS, Area) through its adaptive reward. Furthermore, unlike other methods using the static weight schemes[13,22] or discrete actions, the proposed framework introduces the following:

1. An end-to-end RL framework with a dynamic reward function that rebalances PPA priorities upon a real-time design state.
2. A continuous action space for fine-grained placement control.
3. A direct integration with the commercial EDA tools via a middleware layer.

This enables true co-optimization that adjusts strategy mid-process, akin to an expert designer. Figure 1 shows a comparison between the proposed RL-based co-optimization and traditional sequential optimization.

Methodology

This research offers a framework for integrating reinforcement learning (RL) with traditional EDA tools in a way that is compatible with current design flows and adds adaptive optimization capabilities. Here, it introduces three main parts of the proposed method: the design of the RL framework, the specific implementation of the DDPG agent, and the technique for integrating EDA tools, which allows for practical deployment.

Problem Formulation and Mathematical Modeling

Starting with the physical design optimization problem as a constrained multi-objective optimization issue

Table 1: Comparative analysis of related works in VLSI physical design optimization.

Study	Optimization Focus	AI/ML Technique	PPA Handling	Key Distinction from Auto-PPA
Mirhoseini et al. (2021)[12]	Chip placement	Deep RL (PPO)	Primarily wirelength area	Optimizes proxy metrics; uses discrete placement. Auto-PPA performs direct PPA co-optimization with continuous actions.
Chen et al. (2023)[13]	Power-performance trade-off	Multi-agent RL	Fixed weight scheme	Uses static prioritization. Auto-PPA uses dynamic, state-aware reward balancing.
Kumar et al. (2022)[10]	3D IC placement	Genetic Algorithm	Sequential optimization	High computational cost; prone to local optima. Auto-PPA uses sample-efficient RL for adaptive search.
This paper	Physical design co-optimization	Adaptive DDPG	Dynamic PPA balancing	Integrates adaptive rewards + continuous actions for end-to-end PPA trade-off management.

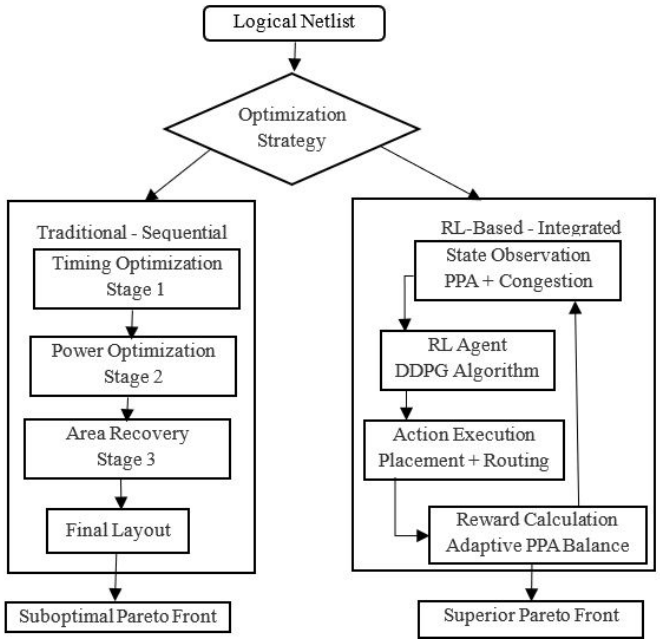

Fig. 1: Comparison of traditional sequential optimization vs proposed RL-based co-optimization.

will provide a solid mathematical basis for the suggested reinforcement learning method. The proposed RL agent's learning objective is defined precisely by this formalization, which also gives the mathematical framework to measure how well a solution performs.

PPA optimization as a constrained Multiobjective problem

The physical design optimization challenge can be formally expressed as a constrained multi-objective optimization problem. Let us define a physical design configuration as **X**, representing the complete set of placement coordinates (x_i, y_i) for all N standard cells, macro blocks, and I/O pads, along with routing configurations R and constraint settings C. The optimization seeks to minimize three competing objective functions simultaneously[18]:

$$\min_{X\in\mathcal{F}}\left[P(X),T(X),A(X)\right]$$

Subject to:

$$g_j(X)\le 0, j=1,2,\ldots,m$$
$$h_k(X)=0, k=1,2,\ldots,p$$

Where:

- $P(X)$: Total power consumption (dynamic + static)
- $T(X)$: Timing violation (primarily Worst Negative Slack)
- $A(X)$: Silicon area utilization
- $\mathcal{F}$: Feasible design space satisfying all constraints
- $g_j(X)$: Inequality constraints (DRC, max capacitance, IR drop limits)
- $h_k(X)$: Equality constraints (clock tree balance, symmetry requirements)

The Pareto optimality condition for this problem states that a design configuration X^* is Pareto optimal if there does not exist another configuration X such that:

$$P(X)\le P(X^*),T(X)\le T(X^*),A(X)\le A(X^*)$$

with at least one strict inequality. The set of all Pareto optimal solutions forms the **Pareto front** in the three-dimensional PPA space.

Mathematical decomposition of objective functions

Each objective function can be further decomposed into measurable physical quantities:

Power Objective:

$$P(X)=P_{\text{dynamic}}(X)+P_{\text{static}}(X)+P_{\text{leakage}}(X)$$
$$P_{\text{dynamic}}(X)=\alpha\cdot C_{\text{total}}(X)\cdot V_{\text{dd}}^2\cdot f_{\text{clock}}$$

Where α is the switching activity factor dependent on placement through netlength, $C_{\text{total}}(X)$ is the total switched capacitance, V_{dd} is the supply voltage, and F_{clock} is the operating frequency.

Timing Objective:

$$T(X)=\max\left(0,-WNS(X)\right)+\lambda\cdot TNS(X)$$
$$WNS(X)=\min_{\forall\text{paths}}\left(Slack_{\text{path}}(X)\right)$$
$$TNS(X)=\sum_{\forall\text{paths}}\max\left(0,-Slack_{\text{path}}(X)\right)$$

Where $WNS(X)$ is the Worst Negative Slack, $TNS(X)$ is the Total Negative Slack, and λ is a weighting factor balancing between fixing the single worst path versus improving overall timing.

Area Objective:

$$A(X)=A_{\text{cell}}(X)+A_{\text{routing}}(X)+A_{\text{white space}}(X)$$

Where $A_{\text{cell}}(X)$ is the total standard cell area, $A_{\text{routing}}(X)$ is the routing channel area estimated from congestion, and $A_{\text{white space}}(X)$ is the unused silicon area that affects manufacturability and yield.

Constraint formulation

The constraint functions $g_j(X)$ and $h_k(X)$ enforce physical realizability and design rules:

Placement Density Constraints:

$$g_{\text{density},b}(X) = \frac{\sum_{i \in \text{bin } b} A_{\text{cell},i}}{A_{\text{bin}}} - D_{\max} \leq 0, \forall b \in \text{placement bins}$$

Where bins are typically 10 × 10 μm regions, and $D_{\max}$ is the maximum allowed placement density (typically 70-80%).

Routing Congestion Constraints:

$$g_{\text{congestion},l}(X) = \frac{\text{Demand}_l(X)}{\text{Capacity}_l} - C_{\max} \leq 0, \forall l \in \text{routing layers}$$

Where $\text{Demand}_l(X)$ is the estimated routing demand based on net length and pin distributions, and Capacity_l is the available routing tracks.

Signal Integrity Constraints:

$$g_{\text{IR drop},n}(X) = V_{\text{dd}} - V_{\text{actual},n}(X) - V_{\text{threshold}} \leq 0, \forall n \in \text{power grid nodes}$$
$$g_{\text{capacitance},k}(X) = C_{\text{load},k}(X) - C_{\max,k} \leq 0, \forall k \in \text{driver cells}$$

Reinforcement learning reformulation

This work reformulate this optimization problem as a Markov Decision Process (MDP) to enable reinforcement learning[23]:

State Space:

$$s_t = \left[\phi_P(X_t), \phi_T(X_t), \phi_A(X_t), \phi_C(X_t), \phi_D(X_t)\right]$$

Where ϕ represents feature extraction functions for power, timing, area, congestion, and design metadata, respectively.

Action Space:

$$a_t = \left[\Delta X_{relative}, \beta_p, \beta_T, \beta_A\right]$$

Where $\Delta X_{\text{relative}}$ are normalized displacement vectors for cell clusters, and β parameters control the adaptive weighting in the next optimization phase.

Transition Dynamics:

$$P(s_{t+1} \mid s_t, a_t) = P_{\text{EDA}}(X_{t+1} \mid X_t, a_t)$$

Representing the stochastic transition in the EDA environment after applying the action a_t to design a state X_t.

Reward Function:

$$r_t = R(s_t, a_t, s_{t+1})$$
$$= w_P \cdot \Delta\tilde{P} + w_T \cdot \Delta\tilde{T} + w_A \cdot \Delta\tilde{A} - \lambda_C \cdot \sum_j \max(0, g_j(X_{t+1}))$$

Where $\Delta\tilde{}$ represents normalized improvements, w parameters are adaptive weights, and λ_c penalizes constraint violations.

The **dynamic weight adaptation** follows:

$$w_i(t) = \frac{\exp\left(-\gamma \cdot \frac{f_i(X_t) - f_i^{\min}}{f_i^{\max} - f_i^{\min}}\right)}{\sum_{j \in P,T,A} \exp\left(-\gamma \cdot \frac{f_j(X_t) - f_j^{\min}}{f_j^{\max} - f_j^{\min}}\right)}$$

Where γ controls adaptation sensitivity, and $f_i^{\min}, f_i^{\max}$ are estimated bounds for the objective i.

Optimization goal and success criteria

The ultimate goal is to find a policy π^* that maximizes the expected cumulative reward[24]:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t r_t \mid \pi\right]$$

Where H is the optimization horizon, and γ is the discount factor balancing immediate versus long-term rewards.

A solution is considered successful if it satisfies:

1. Constraint Satisfaction: All and
2. Pareto Dominance: The solution is not dominated by any other feasible solution in the PPA space
3. Practical Viability: Achieves target frequency with acceptable power and area for the target application

The Mathematical Notation Summary is shown in Table 2.

Each part of the reinforcement learning framework that is described next is tailored to handle this intricate optimization environment, and it takes into account all the details of the mathematical model that is given here.

Reinforcement Learning Framework Architecture

To optimize physical designs, the suggested RL framework uses a continuous control paradigm in which the agent engages with the EDA environment at discrete time intervals. The design is based on the conventional RL model, which includes a state space S, an action space A, a reward function R(s,a,s′), and a transition dynamics P(s′|s,a).[25] However, it employs domain-specific modifications to accommodate VLSI physical design. In every optimization step, the state space records all the design information. This includes geometric data (cell locations, placement density map divided into a

Table 2: Mathematical notation summary.

Symbol	Definition	Typical Range/Units
X	Design configuration	Placement coordinates, routing
$P(X)$	Total power	mW
$T(X)$	Timing violation	ps (negative slack)
$A(X)$	Area utilization	mm^2
$WNS(X)$	Worst Negative Slack	ps
$TNS(X)$	Total Negative Slack	ps
α	Switching activity factor	0.05-0.3
$C_{total}(X)$	Total switched capacitance	pF
D_{max}	Maximum placement density	0.7-0.8
C_{max}	Maximum congestion ratio	0.8-0.9
W_p, W_T, W_A	Adaptive weights	[0,1], sum to 1
γ	RL discount factor	0.9-0.99

16 × 16 grid), timing profile data (worst negative slack, total negative slack per clock domain, critical path distribution), power metrics data (switching activity map, static power density, IR drop hotspots), congestion data (global routing congestion in horizontal/vertical directions, pin density, routing blockage), and design metadata data (technology node parameters, optimization stage, constraint priorities). This multidimensional state representation is encoded into a fixed-size feature vector using a combination of convolutional neural networks for spatial data and multilayer perceptrons for scalar metrics. Unlike prior RL applications in EDA that used discrete action spaces, the proposed formulation defines a continuous action space that better reflects the continuous nature of placement coordinates and constraint parameters, consisting of relative displacement vectors for movable cells in dynamically clustered groups and adaptive weighting parameters for the next optimization phase. A key innovation in PPA trade-off management is the reward function, which penalizes violations of constraints and combines normalized increases in power, time, and area with dynamic weights modified according to design phase and current bottlenecks.

DDPG Agent Implementation

This work utilized the Deep Deterministic Policy Gradient algorithm, an architecture for continuous action spaces that is model-free and off-policy actor-critical.[26] The actor network $\mu(s|\theta^{\mu})$ processes the state vector through three fully connected hidden layers with 512, 256, and 128 neurons, respectively, employing ReLU activation functions and batch normalization between all layers for training stability, ultimately producing an action vector with tanh activation. The critic network $Q(s,a|\theta^{Q})$ uses a dual pathway design to process state and action features independently through fully connected layers, then combines them and processes them further through additional combined layers, finally producing a single Q-value estimate. the training protocol utilizes a replay buffer storing 1,000,000 transitions (, , r_t, $s_{\{t+1\}}$) with Ornstein–Uhlenbeck noise added to actions for exploration (θ = 0.15, σ = 0.2), soft updates for target networks with τ=0.001, a discount factor γ = 0.95 to balance immediate and long-term rewards, and a training frequency where the critic updates every step while the actor updates every 50 steps.

EDA Tool Integration Framework

Four critical tasks are carried out by the RL agent's proprietary Python-based middleware layer, which interfaces with commercial EDA tools. In the first step, it takes data about the state and uses it for routing, analysis, and placement databases, standardizing feature representations from proprietary data formats. Second, as the agent makes judgments, it puts them into action by creating tool-specific Tcl scripts that include commands for incremental placement and optimization. Third, it keeps tabs on convergence by collecting data in real-time, which allows it to assess progress in all PPA aspects at once. In the fourth place, it takes care of training stability, checkpointing and resume capability, making sure that optimization procedures, which can be rather lengthy, are resilient against tool crashes and interruptions. Figure 2 shows the physical design optimization framework's architecture that is based on RL.

Experimental Setup

The proposed RL framework was thoroughly tested using a rigorous experimental approach that included industry-standard benchmarks, suitable baseline comparisons, carefully chosen evaluation metrics, and precise implementation specifications.

Benchmark Circuits and Technology Parameters

This approach was evaluated using six benchmarks from the ISPD 2015 Contest Suite and three industrial designs provided by this paper's collaboration partner to ensure both academic reproducibility and industrial relevance. All designs are implemented in a commercial 7 nm FinFET technology with appropriate standard

cell libraries and memory compilers, with characteristics detailed in Table 3. The selected benchmarks span a wide range of complexities from 250 K to 1.5 M cells, multiple clock domains, and varying target frequencies and power constraints, ensuring comprehensive evaluation across different design scenarios, as shown in Table 3.

Baseline Methods for Comparison

The optimized RL strategy results were compared to those of three state-of-the-art optimization methods that represented different schools of thought in algorithmic thinking. The commercial tool flow follows recommended best practices for PPA optimization utilizing sequential approach and uses industry-standard EDA tools, reflecting present-day industry standard practices. To explore PPA trade-offs, the multiobjective genetic algorithm uses an evolutionary computation approach based on a custom NSGA-II variation. The weighted-sum gradient optimization, which stands for the old school of mathematical optimization, optimizes all stages of the flow at once using constant PPA weights.

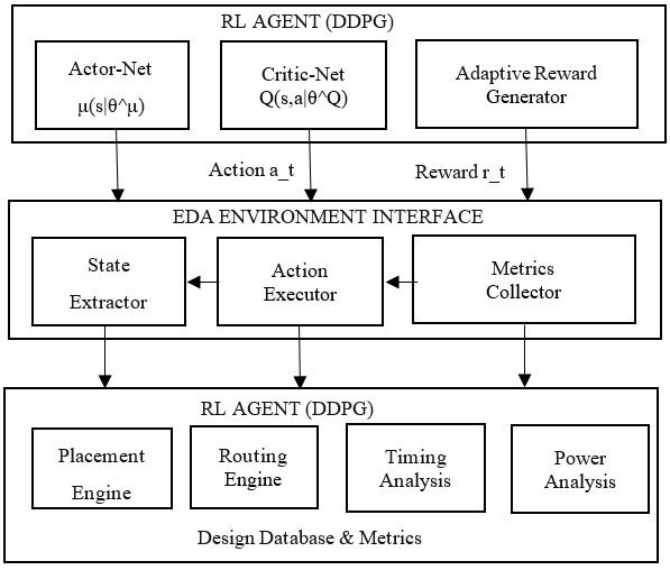

Fig. 2: Architecture of the RL-based Physical Design Optimization Framework

Evaluation Metrics

Various criteria that measure computing efficiency and solution quality are used to assess each methodology. A combined metric for the two most important and conflicting goals is given by the power-performance product, which is computed as Power × Worst Negative Slack. Layout compactness is measured by area efficiency, which is formulated as (Standard Cell Area/Total Chip Area) × 100%. All practical implementation considerations are taken into consideration during runtime, which records the overall time from the wall clock until convergence. The percentage of designs that satisfy all power, time, and design rule constraints is tracked by constraint satisfaction. Pareto front quality uses the hypervolume indicator to compare dominated regions in the three-dimensional PPA space.

Definition of core metrics

The following metrics are used to evaluate solution quality:

- Power-Performance Product (PPP): $PPP = P_{total}\,(mW) \times |\,WNS\,|\,(ps)$. A lower PPP indicates a superior balance between power consumption and timing performance. Improving one metric at a severe expense of the other worsens PPP.
- Area Efficiency: $\text{Area Efficiency} = \frac{A_{standard\ cells}}{A_{total\ chip}} \times 100\%.$ Higher efficiency indicates more compact placement and less wasted silicon.
- Hypervolume Indicator: Measures the volume of the objective space dominated by the obtained Pareto front relative to a reference point. A larger hypervolume indicates a better and more diverse set of PPA trade-offs.

Training and Implementation Details

The details of the implementation guarantee repeatable outcomes that are reflective of actual deployment scenarios. Training takes about 72 hours on a 32-core

Table 3. Benchmark circuit characteristics.

Circuit	Cells (K)	Nets (K)	Clock Domains	Target Frequency	Power Constraint
SuperBlue1	250	275	3	2.1 GHz	250 mW
SuperBlue5	420	480	5	1.8 GHz	420 mW
SuperBlue7	680	720	4	1.5 GHz	650 mW
Industrial_A	1250	1350	8	3.0 GHz	1.1 W
Industrial_B	850	920	6	2.5 GHz	850 mW
Industrial_C	1500	1620	12	2.2 GHz	1.4 W

server and covers 500,000 steps on SuperBlue benchmarks, while inference takes about 2–4 hours per design, depending on complexity. A 32-core Xeon server and an NVIDIA A100 GPU are part of the hardware infrastructure used for RL training and EDA execution, respectively. A bespoke middleware enables compatibility between the software components, which include TensorFlow 2.8, Python 3.9, and commercial EDA tools version 2022, Table 4 shown detailed experimental configuration.

Results and Analysis

The experimental results show that the suggested RL framework is consistently better than the alternatives, especially when it comes to optimizing PPAs simultaneously and being computationally efficient.

PPA Optimization Comparison

The proposed RL agent showed superior optimization capability across all benchmark circuits, achieving better trade-offs between competing objectives compared to baseline methods. Table 5 shows the PPA finding comparison across optimization technologies. For the SuperBlue5 benchmark, Auto-PPA gets a PPP of 3258 mW·ps, 45% less than the PPP value 5970 mW·ps of commercial tool (Table 5). This notable decrease is the result of effective co-optimization, since improved timing and power were optimized simultaneously (Power: 362 vs 398 mW; WNS: −9 vs −15 ps).

When compared to the commercial tool flow, the RL agent saves 12.4% in average power, 41.7% in timing improvement (WNS), and 7.8% in area. Among these competing limitations, the power-performance product stands out with an average improvement of 18.7%, suggesting better balancing. The 3D Pareto Front Analysis in PPA Space for the SuperBlue7 Benchmark is displayed in Figure 3. The RL agent achieves points closer to the ideal origin (low power, zero WNS, small area), demonstrating superior multiobjective optimization.

Table 4: Detailed experimental configuration.

Category	Parameter	Value/Specification	Purpose/Note
RL Agent (DDPG)	Actor Learning Rate	1.00E-04	Adam optimizer
	Critic Learning Rate	1.00E-03	Adam optimizer
	Discount Factor (γ)	0.95	For cumulative reward
	Replay Buffer Size	1,000,000 transitions	Experience replay
	Batch Size	64	For network updates
	Target Update (τ)	0.001	Soft target network update
	Exploration Noise	Ornstein-Uhlenbeck (θ = 0.15, σ = 0.2)	For action space exploration
Neural Networks	Actor Architecture	512 → 256 → 128 (ReLU)	Outputs: continuous action vector
	Critic Architecture	State: 512 → 256; Action: 128; Combined: 256 → 128 (ReLU)	Outputs: Q-value
EDA Environment	Placement & Routing Tool	Synopsys Fusion Compiler™ 2022.12	Industry-standard P&R flow
	Timing/Power Analysis	Synopsys PrimeTime® 2022.12	For signoff-quality metrics
	Technology Node	7nm FinFET PDK	Commercial process design kit
Hardware & Software	Training Server	32-core Intel Xeon, 256GB RAM, NVIDIA A100 80GB GPU	For RL training
	Inference Platform	16-core Intel Xeon, 128GB RAM	For running optimized agent
	Operating System	Ubuntu 20.04 LTS	
	Random Seed	42	For the reproducibility of all stochastic processes

Table 5: PPA results comparison across optimization methods.

Circuit	Method	Power (mW)	WNS (ps)	Area (mm²)	PPP (mW·ps)	Area Eff.
SuperBlue5	Commercial	398	−15	0.142	5970	71.3%
SuperBlue5	MOGA	385	−22	0.138	8470	73.1%
SuperBlue5	RL (Ours)	362	−9	0.131	3258	76.8%
Industrial_B	Commercial	872	−28	0.285	24416	68.9%
Industrial_B	MOGA	845	−35	0.279	29575	70.2%
Industrial_B	RL (Ours)	821	−19	0.266	15599	73.4%

Runtime Efficiency and Convergence Analysis

By reducing the iteration count and optimizing the exploration of the solution space, the suggested RL technique shows faster overall convergence, even when neural network inference has an initial overhead. Values of normalized metrics throughout 25 optimization iterations for three techniques are displayed in Figure 4 and Table 6, respectively.

The RL agent improves in tandem with each of the three metrics, however the baseline approaches display the classic "seesaw" effect, whereby boosting one parameter negatively impacts the others. Despite computing complexity per iteration, the RL method decreases total runtime by 35% and averages a 42% reduction in optimization iterations compared to the commercial flow.

Ablation Studies and Component Analysis

Isolating the contribution of different framework components and validating design decisions required a comprehensive ablation study. Three implementations on the Industrial C circuit were compared to assess reward function variants: fixed equal weights, manual stage-based weights, and the suggested adaptive RL weights. Table 7 displays the results of the Industrial C Circuit Reward Function Ablation Study:

The suggested adaptive method outperformed the best fixed-weight alternative by 11.4% in terms of Pareto front hypervolume, with constraint satisfaction at 96%, compared to 92 and 87% for for manual and fixed weights. To evaluate the effect of the action space design, a comparison was made between the suggested continuous approach and discrete alternatives, such as grid-based movement and parameterized macroactions. The results showed that continuous control produced 23% higher quality solutions than discrete alternatives, proving the importance of fine-grained control in physical design optimization.

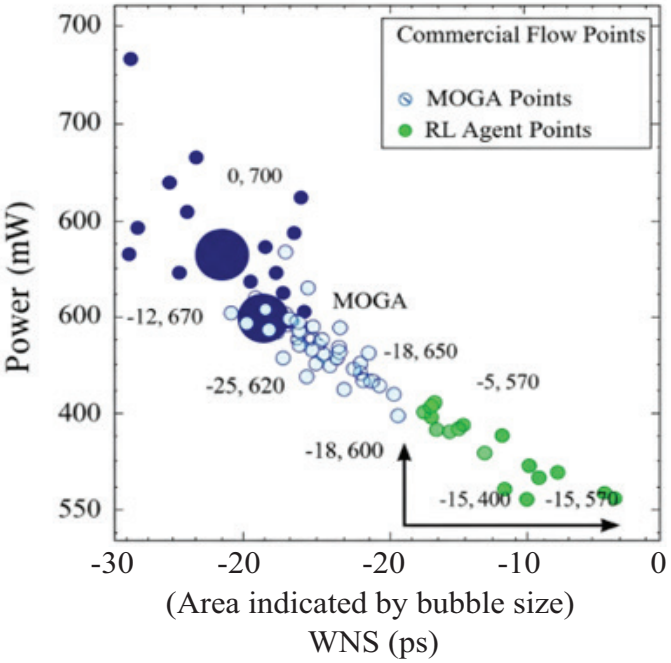

Fig. 3: Pareto Front Comparison in 3D PPA Space for SuperBlue7 Benchmark.

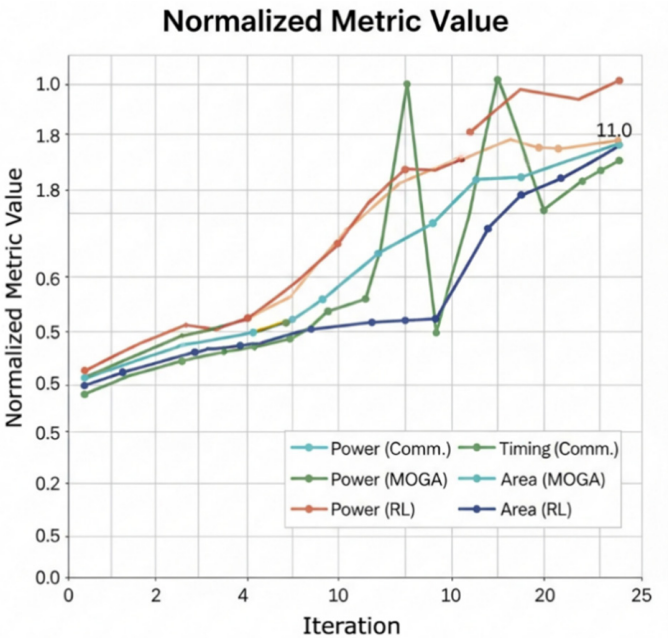

Fig. 4: Optimization Convergence Comparison for Industrial_A Design.

Table 6: Data for Figure 4 – Convergence comparison for Industrial_A.

Iteration	Power_Norm (Comm)	Timing_Norm (Comm)	Area_Norm (Comm)	Power_Norm (MOGA)	Timing_Norm (MOGA)	Area_Norm (MOGA)	Power_Norm (RL)	Timing_Norm (RL)	Area_Norm (RL)
0	1	1	1	1	1	1	1	1	1
1	0.982	0.85	0.995	0.975	0.82	0.99	0.965	0.78	0.985
2	0.965	0.75	0.99	0.952	0.71	0.982	0.935	0.62	0.972
3	0.95	0.68	0.985	0.93	0.635	0.975	0.905	0.52	0.96
4	0.935	0.62	0.98	0.91	0.58	0.968	0.88	0.45	0.95
5	0.92	0.57	0.975	0.892	0.535	0.962	0.86	0.4	0.942
6	0.905	0.53	0.97	0.875	0.5	0.957	0.842	0.36	0.935
7	0.892	0.495	0.965	0.86	0.47	0.952	0.825	0.33	0.928
8	0.88	0.465	0.96	0.845	0.445	0.947	0.81	0.305	0.922
9	0.868	0.44	0.955	0.832	0.425	0.943	0.796	0.285	0.917
10	0.858	0.42	0.95	0.82	0.41	0.94	0.785	0.27	0.913
11	0.848	0.405	0.945	0.81	0.4	0.938	0.775	0.26	0.91
12	0.84	0.395	0.94	0.802	0.395	0.936	0.768	0.255	0.908
13	0.833	0.388	0.935	0.795	0.393	0.935	0.762	0.253	0.907
14	0.828	0.385	0.93	0.79	0.392	0.935	0.758	0.252	0.906
15	0.825	0.384	0.925	0.787	0.392	0.935	0.756	0.252	0.906
16	0.823	0.383	0.92	0.785	0.391	0.934	0.755	0.251	0.905
17	0.822	0.383	0.915	0.784	0.391	0.934	0.754	0.251	0.905
18	0.821	0.382	0.91	0.783	0.391	0.934	0.753	0.251	0.905
19	0.82	0.382	0.905	0.782	0.391	0.934	0.753	0.251	0.905
20	0.82	0.382	0.9	0.782	0.391	0.934	0.753	0.251	0.905
21	0.819	0.382	0.895	0.782	0.391	0.934	0.753	0.251	0.905
22	0.819	0.382	0.89	0.781	0.391	0.933	0.752	0.251	0.905
23	0.819	0.382	0.885	0.781	0.391	0.933	0.752	0.251	0.905
24	0.819	0.381	0.88	0.781	0.391	0.933	0.752	0.251	0.905
25	0.819	0.381	0.875	0.781	0.391	0.933	0.752	0.251	0.905

Generalization and Scalability Analysis

A test was conducted on the agent trained on SuperBlue benchmarks to assess its generalization capacity. The agent was then tested on unseen industrial designs without fine-tuning. Table 8 compares three situations regarding generalization performance on unseen circuits.

Starting from the beginning, adjusting with 10% of the training stages, and transferring without taking any shots. By successfully generalizing learnt optimization tactics, the pretrained agent improved the power-performance product by 11% and accelerated convergence by 22% even on unknown circuits. The computational cost of applying the proposed approach to new design families is greatly reduced by this transfer capability. The pretrained agent gives rapid benefits without extensive retraining.

Discussion and Future Work

There are several significant benefits to the suggested RL strategy over more conventional approaches, and these explain why it outperforms the competition. To avoid the "whack-a-mole" dilemma, where improving one measure lowers another, the RL agent takes into account all PPA ramifications of each action at the same time, in contrast to sequential techniques. The agent can emulate the techniques of expert designers with more consistency and computing efficiency and thanks to the dynamic incentive weighting, which allows it to

Table 7: Reward function ablation study (Industrial_C Circuit).

Reward Variant	Final PPP	Hypervolume	Constraint Met
Fixed equal weights	28740	0.712	87%
Manual stage-based weights	25420	0.765	92%
Adaptive RL weights (Ours)	21850	0.823	96%

Table 8. Generalization performance on unseen circuits.

Circuit	Transfer Learning	PPP Relative	Runtime Relative
Industrial_D	None (train from scratch)	1.00×	1.00×
Industrial_D	Fine-tune (10% steps)	0.94×	0.82×
Industrial_D	Zero-shot (Ours)	0.89×	0.78×

prioritize timing convergence early, power recovery, and area optimization in that order. Moreover, unlike heuristic algorithms, the agent learns from its mistakes and has a "design intuition" that helps avoid underperforming areas of the solution space.

Besides algorithmic performance, there are a number of implementation factors that need to be considered for actual industrial use. A middleware layer is required for tool integration robustness to manage version changes and variances in database formats across different releases of EDA tools. Although RL training is inherently stochastic, inference for production designs must yield deterministic results, necessitating meticulous control over random seeds and startup states. Hierarchical RL techniques provide a promising solution to the problem of appropriately encoding complicated hierarchical constraints in state representations without leading to excessive dimensionality increase in real-world designs.

However, there are some important caveats to the proposed technique that point to fruitful avenues for further study. Even though the initial training uses a lot of computer resources, the cost spreads out over several designs. To speed up training for new technology nodes, future research should look into meta-learning techniques. For designs with more than 5 million cells, hierarchical or distributed RL methods may be necessary to keep scalability. An expansion of the framework to incorporate 3D-IC and advanced packaging technologies would meet the demands of the industry going forward. Ultimately, by removing artificial barriers in existing design procedures, more PPA gains could be unlocked by broadening the optimization scope to incorporate combined optimization with logic synthesis.

Conclusion

To optimize power, performance, and space in VLSI physical design all at once, this study introduces a new RL paradigm. The proposed strategy successfully traverses the complicated trade-off landscape provided by conventional sequential approaches, by utilizing a DDPG agent with adaptive reward weighting and a continuous action space. By incorporating intelligent, adaptive optimization capabilities into traditional design flows, the framework guarantees practical applicability and integrates with commercial EDA tools through a middleware layer.

Experimental results demonstrating consistent improvements across industry-standard benchmarks, with an average 18.7% enhancement in power-performance product, 12.3% reduction in area, and 35% faster convergence compared to state-of-the-art commercial tools. Research on ablation has shown that continuous action spaces and adaptive reward mechanisms are crucial, and results from generalization testing point to potential applicability to newly discovered circuit types. As the complexity of optimization grows due to technological scaling, these developments tackle important issues in current VLSI design.

The proposed framework represents a significant step toward intelligent, autonomous EDA systems that could adapt to change design requirements and technological constraints. Such AI-driven optimization approaches will be vital to attain the optimal PPA trade-offs within realistic design limits as the complexity of VLSI designs is increasing rapidly alien with the new technologies. Improving generalization using meta-learning approaches and expanding the concept to 3D-IC designs are the main goals of future development.

References

1. Zhou, Z. (2022). Machine learning-based techniques for routing interconnects in very large scale integrated (VLSI) circuits (Master's thesis). University of British Columbia.
2. Jenila, R., & Naidu, K. J. (2025). A survey on learning-based PnR optimization for VLSI physical design

automation. IEEE Access, 13, 195953–195974. https://doi.org/10.1109/ACCESS.2025.3631956
3. Sapatnekar, S. S., Kumar, A., Lo, H., Moy, W., Prova, N. S., & Zeng, Z. (2025). Towards designing and deploying Ising machines. In Proceedings of the International Symposium on Physical Design (pp. 1–1). https://doi.org/10.1016/j.vlsi.2022.05.003
4. Szentimrey, H., Al-Hyari, A., Foxcroft, J., Martin, T., Noel, D., Grewal, G., et al. (2020). Machine learning for congestion management and routability prediction within FPGA placement. ACM Transactions on Design Automation of Electronic Systems, 25(5), 1–25. https://doi.org/10.1145/3373269
5. Kulkarni, A. M., & Chopde, A. (2024). Physical design: Methodologies and developments. arXiv. https://doi.org/10.48550/arXiv.2409.04726
6. Mutar, H. A., Najeeb, S. M. M., Aldabag, M. L., & Salim, H. T. (2026). Hybrid neural network—Genetic algorithm framework for EEG and ECG signal classification. Journal of Medical Computing. https://doi.org/10.53759/7669/jmc202606021
7. Bazgan, C., Ruzika, S., Thielen, C., & Vanderpooten, D. (2022). The power of the weighted sum scalarization for approximating multiobjective optimization problems. Theory of Computing Systems, 66(1), 395–415. https://doi.org/10.1007/s00224-021-10066-5
8. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2022). Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 16000–16009).
9. Li, J., Xu, S., Zheng, J., Jiang, G., & Ding, W. (2024). Research on multi-objective evolutionary algorithms based on large-scale decision variable analysis. Applied Sciences, 14(22), 10309. https://doi.org/10.3390/app142210309
10. Qiu, Y., Xing, Y., Zheng, X., Gao, P., Cai, S., & Xiong, X. (2023). Progress of placement optimization for accelerating VLSI physical design. Electronics, 12(2), 337. https://doi.org/10.3390/electronics12020337
11. Mutar, H. A., Yalwi, I. K., Mezaal, Y. S., ALRikabi, H. T. S., Alaidi, A. H. M., ALRubeei, I. R. N., et al. (2025). Investigation of AI with OpenCV-Python for detecting diabetes. In Duraković, B., Almisreb, A.A., & Šutković, J. (Eds.), Recent trends and applications of soft computing in engineering (pp. 217–231). Springer Nature. https://doi.org/10.1007/978-3-031-82881-2_14
12. Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J.W., Songhori, E., Wang, S., et al. (2021). A graph placement methodology for fast chip design. Nature, 594, 207–212. https://doi.org/10.1038/s41586-021-03544-w
13. Michael, P., & Jackson, K. (2025). Advancing scientific discovery: A high performance computing architecture for AI and machine learning. Journal of Integrated VLSI, Embedded and Computing Technologies, 2(2), 18–26. https://doi.org/10.31838/JIVCT/02.02.03
14. Zang, Z., Song, Y., Ling, B. W.-K., Wang, A., & Yang, F. (2025). The dawn of agentic EDA: A survey of autonomous digital chip design. arXiv. https://doi.org/10.48550/arXiv.2512.23189
15. AlRubeei, I. R., Idi, S. N., Alsammak, I. L. H., AlRikabi, H. T., Mutar, H. A., & Alaidi, A. H. M. (2025). Using artificial intelligence for enhancement of solar cell efficiency in the south of Iraq.
16. Abdul-Hussein, M. K., Mutar, H. A., AlRikabi, H. T. S., AlRubeei, I. R. N., Svyd, I., & Mezaal, Y. S. (2024). Investigation of an interference communication system to overcome cheating based on IoT techniques. In Proceedings of the Conference on Recent Trends and Applications of Soft Computing in Engineering (pp. 69–82). Springer. https://doi.org/10.1007/978-3-031-82881-2_5
17. Khetarpal, V., Gupta, L., Dhand, R., & Sharma, P. (2024). Machine learning techniques for VLSI circuit design: A review. In Abraham, A., et al. (Eds.), Intelligent systems design and applications (pp. 191–199). Springer Nature.
18. Uvarajan, K. P. (2026). Constraint-guided data-driven learning models for high-dimensional nonlinear control systems. Journal of Scalable Data Engineering and Intelligent Computing, 9–16.
19. Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J., Songhori, E., Wang, S., et al. (2020). Chip placement with deep reinforcement learning. arXiv. https://doi.org/10.48550/arXiv.2004.10746
20. Velliangiri, A. (2025). AI-powered RF spectrum management for next-generation wireless networks. National Journal of RF Circuits and Wireless Systems, 2(1), 21–29.
21. Vecerik, M., Sushkov, O., Barker, D., Rothörl, T., Hester, T., & Scholz, J. (2019). A practical approach to insertion with variable socket position using deep reinforcement learning. In Proceedings of the IEEE International Conference on Robotics and Automation (pp. 754–760). https://doi.org/10.1109/ICRA.2019.8794074
22. Joshua Reginald, P. (2025). Scalable distributed learning and control pipelines for large-scale nonlinear actuation systems. SECITS Journal of Scalable Distributed Computing and Pipeline Automation, 1–8.
23. Kim, T. (2025). Physical design challenges for design technology co-optimization. In Proceedings of the International Symposium on Physical Design (pp. 20–21). https://doi.org/10.1145/3698364.3709119
24. Chung, S., Seo, H., Cho, H., Choi, K., & Kim, T. (2024). Optimal layout synthesis of multi-row standard cells for advanced technology nodes. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (pp. 1–8).
25. Abdalla, R., Hollstein, W., Carvajal, C. P., & Jaeger, P. (2023). Actor–critic reinforcement learning–led decision-making in energy systems optimization: Steam injection optimization. Neural Computing and Applications, 35(22), 16633–16647. https://doi.org/10.1007/s00521-023-08537-6
26. Lillicrap, T. P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. arXiv. https://doi.org/10.48550/arXiv.1509.02971