Cross-Stage Feedback Learning for Power, Performance, And Area Optimization in Open-Source ASIC Physical Design

Shubhashri G. Joshi[1], Shrikant J. Honade[2], Ganesh G. Patil[3], Ganesh B. Dongre[4]
[1,2,3,4]*Department of Electronics Engineering, CSMSS Chh. Shahu College of Engineering, Chhatrapati Sambhajinagar, India*

***Abstract*—Accurate pre-implementation estimation of power, performance, and area (PPA) remains a major bottleneck in application-specific integrated circuit (ASIC) design, because register-transfer level (RTL)-to-GDSII flows such as OpenLane require computationally expensive iterative synthesis and place-and-route executions. This paper presents CSF-MORL, a cross-stage feedback-driven learning framework for pre-implementation PPA prediction and design parameter optimization, targeted at embedded, reconfigurable, and IoT-oriented ASIC design flows. Compact structural features extracted from RTL descriptions drive a multi-model pipeline of random forest regressors for die area and total power and a binary pass/fail classifier, while a reinforcement learning agent refines clock period, core utilization, and placement density using post-route feedback. A per-design normalization scheme decouples prediction accuracy from design-size variability, and a design-similarity transfer mechanism retargets trained surrogates to previously unseen RTL without retraining. The framework achieves 31.3% average area prediction error and accurate timing pass/fail classification on all four unseen designs, with zero design-rule violations across all nine, and reduces design space exploration time by up to three orders of magnitude. It was validated on nine digital designs and 193 OpenLane runs on the SkyWater 130 nm process design kit.**

***Index Terms*—ASIC physical design; Design space exploration; Machine learning; multi-objective optimization; OpenLane; Reinforcement learning**

I. INTRODUCTION

Physical design is one of the most challenging tasks in modern very large-scale integration (VLSI) systems because of the increasing complexity of integrated circuits [1]. Achieving optimal power, performance, and area (PPA) while achieving design closure becomes more difficult as technology scales. Conventional application-specific integrated circuit (ASIC) physical design flows like OpenLane [2] explore some key parameters such as clock period, core utilisation and placement density iteratively to satisfy design constraints on open process design kits such as the SkyWater 130 nm platform [3]. These flows are effective but computationally and time-consuming especially for user-defined designs with a variety of structural features [2], [3].

Early-stage PPA estimates and design space exploration can be accelerated by machine learning (ML) techniques [4], [5]. ML models can approximate physical design results without complete tool execution [6], [7] by learning from past design data. However, most traditional approaches treat prediction and optimisation as distinct tasks [9], [10], and current methods often show poor generalisation across heterogeneous designs, limited scalability, and an inability to reflect the intrinsically multi-objective character of physical design optimisation [8].

Beyond algorithmic considerations, the problem addressed in this work is directly relevant to the design of reconfigurable and embedded systems. Open-source flows such as OpenLane [2] and OpenROAD [11], combined with open process design kits [3], are democratizing custom silicon and enabling small teams to prototype embedded and internet of things (IoT) ASICs that were previously accessible only to large semiconductor companies. In this context, a lightweight surrogate that predicts PPA from an RTL description alone supports reconfigurable design space exploration across heterogeneous RTL inputs, and its

second-scale inference and small memory footprint allow it to run in resource-constrained environments.

Considerable prior work has applied learning-based methods to physical design. Reinforcement learning (RL) has emerged as a powerful paradigm for sequential decision making in chip design: Mirhoseini et al. [12] demonstrated RL-based placement and floor planning, broader directions for ML in physical implementation were surveyed by Kahng [13], Agnesina et al. [14] proposed a general framework for tool parameter optimization with deep RL, and AutoDMP [15] and RTL-MP [16] addressed macro placement. However, most existing RL-based approaches are limited to single-stage optimization and do not incorporate feedback from later stages such as routing. In parallel, ML-based congestion and routability prediction has been studied using convolutional networks [17], routability predictors such as RouteNet [18], graph-based placement models [19], congestion predictors for large-scale devices [20], detailed-routing violation classifiers [21], and ML-guided routability optimization [22]; these predictors are typically static and lack mechanisms to influence earlier stages. From this literature, three major limitations can be identified: i) lack of cross-stage optimization; ii) absence of routing-aware feedback mechanisms; and iii) limited support for unified multi-objective optimization.

To address these gaps, this paper proposes CSF-MORL, a cross-stage feedback-driven learning framework for efficient PPA estimation and optimization in OpenLane ASIC physical design. Unlike prior single-stage or static approaches, the framework combines feature-based design characterization with a multi-model learning strategy and a reinforcement learning agent whose reward is computed from post-route signoff metrics. A per-design calibration and normalisation method ensures robustness across various scale designs. This work's primary contributions are: i) a cross-stage feedback mechanism where routing-stage outcomes determine parameters acting at synthesis and floorplanning; ii) a multi-model pipeline with independent random forest regressors for die area and total power along with a global pass/fail classifier, trained on 193 OpenLane runs across five benchmark designs; iii) an area-aware design similarity metric for choosing the best per-design model for unseen RTL; iv) end-to-end validation on four previously unseen designs that show 31.3% average area prediction error and accurate timing pass/fail classification; and v) an automated configuration generator generating files from RTL input that are compatible with OpenLane.

The rest of the paper is organised as follows. Section 2 describes the proposed framework and its formal problem formulation. Section 3 details in depth the methodology, dataset creation and experimental setting. Results and discussion are presented in section 4. Section 5 concludes the paper.

II. THE PROPOSED CSF-MORL FRAMEWORK

Figure 1 shows the architecture of CSF-MORL. The framework accepts a Verilog RTL description as its sole input and produces an OpenLane-compatible configuration file together with predicted PPA metrics. It comprises three interacting components: i) a surrogate prediction pipeline built from per-design random forest models (section 3.4); ii) a design-similarity transfer mechanism that selects an appropriate surrogate for previously unseen RTL (section 3.5); and iii) a reinforcement learning-based parameter tuning agent that refines configurations using feedback from executed OpenLane flows (section 3.6). Components i) and ii) execute in seconds without invoking the physical design tools, while component iii) closes the loop against real flow results.

2.1. Design space exploration problem

Given an RTL description V of a digital circuit and a target technology library L [1], [3], the design space exploration problem seeks physical design parameters $\theta^* = (clk^*, u^*, d^*)$ that minimize a weighted multi-objective cost:

$$J(\theta) = w_1 A(\theta) + w_2 P(\theta) + w_3 |W(\theta)| \quad (1)$$

where A, P, and W denote die area, total power, and worst negative slack (WNS) respectively, and w_1, w_2, w_3 are weighting coefficients.

The admissible parameter space is

$$\Theta = \{ clk \in [8, 15] \text{ ns}, u \in [0.30, 0.50], d \in [0.30, 0.50] \} \quad (2)$$

during agent-driven exploration, and is widened to $clk \in [1, 25]$ ns, $u \in [0.40, 0.75]$, $d \in [0.50, 0.75]$ when configurations are generated directly by the surrogate pipeline.

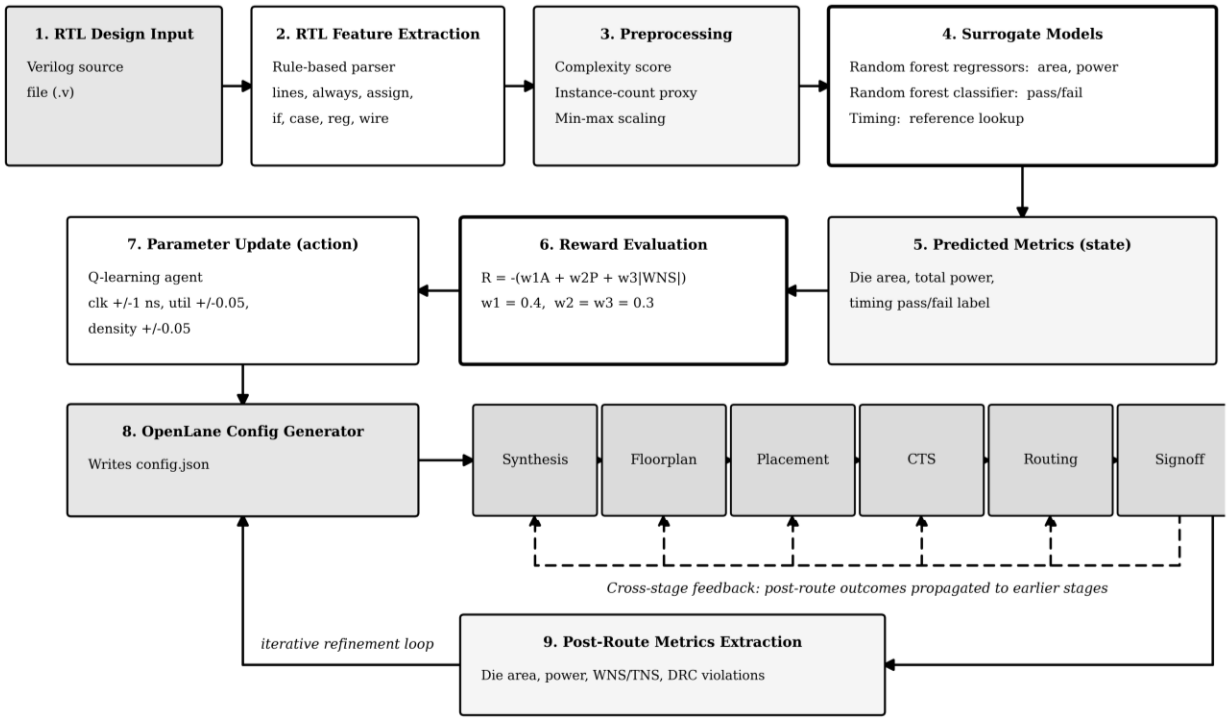

Figure 1. Overall architecture of the proposed CSF-MORL framework, showing the surrogate prediction pipeline (stages 1–4), the reward-driven parameter update loop (stages 5–8), and the cross-stage feedback path from post-route metrics extraction (stage 9) back to earlier design stages

The training corpus was produced by sweeping this space using Latin hypercube sampling [24], as detailed in section 3.7.

2.2. State, action, and reward formulation

The parameter tuning problem is cast as a Markov decision process. The state at iteration t is the tuple s_t = (design, clk, u, d, WNS, P, A), discretized by rounding each continuous component to a fixed precision so that the state space remains enumerable, and the action set is discrete and single-step, $\mathcal{A}$ = {inc_clk, dec_clk, inc_util, dec_util, inc_den, dec_den}, where clock actions adjust the period by ±1 ns and utilization and density actions by ±0.05, with all updates projected back into Θ after application. The reward returned after a completed flow, with $w_1 = 0.4$ and $w_2 = w_3 = 0.3$, is

$$R = -(w_1 A + w_2 \cdot 10^3 P + w_3 \cdot 10^2 |WNS|) \quad (3)$$

The scaling constants 10^3 and 10^2 bring power (in watts) and slack (in nanoseconds) onto a magnitude comparable with area (in μm^2), so that no single objective dominates the weighted sum. A flow that fails to complete returns a large negative constant, which encodes infeasibility as an absorbing penalty and steers the agent away from parameter regions that do not produce a manufacturable result.

2.3. Novelty with respect to prior learning-based methods

CSF-MORL differs from prior RL-based placement and floorplanning frameworks [12], [15], [16] in three respects. First, prior approaches optimize a single physical design stage and evaluate their reward from that stage alone, whereas the reward of (3) is computed from post-route signoff metrics extracted after the complete OpenLane flow, so that routing-stage outcomes govern parameters acting at synthesis and floorplanning. Second, prior work requires a full tool invocation for every candidate configuration, whereas coupling the agent to a surrogate model reduces the cost of evaluating a candidate from minutes to milliseconds, and a design-similarity mechanism transfers a trained surrogate to previously unseen RTL without retraining, allowing a manufacturable configuration to be generated for an arbitrary user-defined circuit from its Verilog source alone. Third, die areas in the training corpus span three orders of magnitude, from approximately 2.2×10^3 to 2.1×10^6

μm^2; normalizing each target to its per-design maximum decouples model accuracy from absolute design size, which is a dominant cause of poor cross-design generalization in earlier ML-based predictors [8].

III. METHOD

This section describes the framework in execution order, from RTL feature extraction through to the evaluation protocol. All parameter values, tool versions, and procedures required to reproduce the study are stated explicitly.

3.1. RTL feature extraction

Structural features are extracted by a rule-based parser implemented in Python. Comments are stripped by regular expression, after which the parser performs a single lexical pass and counts the number of non-empty source lines, always blocks, continuous assign statements, if statements, case statements, port declarations, register declarations, and wire declarations.

The module identifier is recovered from the module declaration. This yields the raw descriptor

$$X_{RTL} = [\, n_{lines}, n_{always}, n_{assign}, n_{if}, n_{case}, n_{reg}, n_{wire} \,] \quad (4)$$

Extraction requires no synthesis, no elaboration, and no testbench, and completes in milliseconds. From this descriptor two derived quantities are computed. A scalar complexity score $S = n_{lines} + 5n_{always} + 2n_{if} + 3n_{case} + 1.5n_{reg}$ assigns the design to the simple ($S < 50$), moderate ($50 \le S < 150$), or complex ($S \ge 150$) class.

An estimated instance counts

$$\hat{N}_{inst} = \max(50, 50n_{lines} + 200n_{always} + 30n_{reg} + 20n_{if}) \quad (5)$$

provides a pre-synthesis proxy for post-synthesis gate count. The coefficients of this score and of (5) were calibrated against observed instance counts in the training corpus and are held fixed thereafter. The estimated standard-cell area follows as $\hat{A}_{inst} = 1.8\,\hat{N}_{inst}$ μm^2, using the mean cell footprint of the sky130 standard cell library.

3.2. Feature encoding and preprocessing

The surrogate regressors operate on a compact four-dimensional vector combining the structural proxy with the three swept configuration parameters:

$$X = [\, \hat{N}_{inst}, clk, u, d] \quad (6)$$

The regressors therefore learn the response of area and power to the configuration parameters, conditioned on design scale through $\hat{N}_{inst}$.

The RTL descriptor is used instead for model selection (section 3.5) and by the global pass/fail classifier, which is trained across all designs and receives the eleven-dimensional vector

$$X_{clf} = [\, n_{lines}, n_{always}, n_{assign}, n_{if}, n_{case}, \hat{N}_{inst}, \hat{A}_{inst}, clk, u, d, \delta] \quad (7)$$

where δ is the label-encoded design identifier. Categorical variables are one-hot or label encoded as appropriate, and numerical features are min-max scaled with statistics computed on the training partition only, so that no information from validation folds or from held-out designs enters training.

3.3. Design-aware normalization

Target metrics vary by three orders of magnitude across the corpus (Table 2). Each target is therefore normalized to its per-design maximum before training, $Y_{norm} = Y / Y_{max}^{(design)}$, and the inverse transform $Y = Y_{pred} \cdot Y_{max}^{(design)}$ is applied at inference. For an unseen design the scale factor is inherited from the matched training design identified in section 3.5.

3.4. Surrogate model construction

The surrogate layer comprises eleven trained artifacts: five per-design random forest regressors for die area, five per-design random forest regressors for total power, and one global random forest classifier for pass/fail prediction, together with a label encoder for the design identifier.

Random forest was selected for its robustness to the run-to-run variation introduced by nondeterministic placement, its resistance to overfitting on small tabular partitions, and its low inference cost, which permits interactive use during early-stage exploration. Table 1 lists the complete configuration.

All settings not shown retain scikit-learn 1.8.0 defaults, and a fixed random seed of 42 is used for every estimator so that all reported figures are exactly reproducible.

The regressors consider all four input features at every split, so the ensemble's stochastic component derives entirely from bootstrap resampling; the decision tree baseline is restricted to depth five and is therefore not depth-matched to the unrestricted forest.

Table 1. Surrogate model configuration

Setting	Regressors (area, power)	Classifier (pass/fail)
Estimator	Random forest regressor	Random forest classifier
Trees / max depth	300 / unrestricted	300 / unrestricted
Min samples per split, leaf	2, 2	2, 1
Features per split	All (4)	$\sqrt{11} \approx 3$
Input dimensionality	4, (6)	11, (7)
Models trained	5 area, 5 power	1 global
Bootstrap, seed, CV	Enabled, 42, 5-fold	Enabled, 42, 5-fold
Baselines	Linear regression; decision tree (max depth 5)	Linear regression; decision tree (max depth 5)
Scaling / software	Min-max (training partition); scikit-learn 1.8.0 [25], Python 3.12	Min-max (training partition); scikit-learn 1.8.0 [25], Python 3.12

Cross-validation uses five folds throughout, the fold count being set adaptively as min (5, max (2, ⌊n/4⌋)) so that no fold falls below a usable size; with per-design partitions of between 23 and 52 runs these resolves to five folds for every area and power model, and the same protocol is applied to the global classifier over the full 193-run corpus.

Timing and routing wirelength are not modelled uniformly across the corpus, for reasons that are a property of the data rather than of the estimator. Setup slack is recorded only for runs exhibiting violations, yielding 7 usable samples for the ALU, 16 for the MAC, and 26 for the UART, and none for the counter and FIFO, which met timing at every sampled configuration. The three timing regressors that could be trained all achieve negative cross-validated R^2 (−0.167, −0.854, and −1.210 respectively), indicating no predictive capability beyond the sample mean. At inference time timing is therefore obtained by reference lookup, taking a conservative percentile of the observed slack distribution over the successful runs of the matched training design, which grounds the pass/fail decision in measured signoff outcomes for a structurally comparable circuit. Routing congestion and wirelength are not modelled at all, as the corpus records neither quantity; graph neural networks operating on the synthesised netlist are identified as the appropriate remedy in section 5.

Feature importance, computed as the mean impurity decrease across each ensemble, shows that for the area models the estimated instance count of (5) dominates overwhelmingly, accounting for 0.996 of the total importance in the ALU model against 0.003, 0.002, and <0.001 for clock period, utilization, and placement density. This is physically consistent, since die area is determined primarily by the number of synthesised standard cells. The area surrogate is therefore effectively a learned calibration of the instance-count proxy against measured post-route area rather than a model of the configuration response, and its accuracy is bounded by that of the proxy.

3.5. Design-similarity transfer and configuration generation

For RTL not present in the training corpus, the framework selects the surrogate whose training design is structurally closest to the query, combining two criteria: the distance between the estimated die area of the query, from (5), and the mean observed die area $\bar{A}_j$ of each training design j, measured on a logarithmic scale to accommodate the three-decade spread; and the Euclidean distance between range-normalized RTL descriptors.

The matched design is

$$j^{*} = \operatorname{argmin}_j \left[0.8 \left| \log(1 + \hat{A}) - \log(1 + \bar{A}_j) \right| + 0.2 \left\| \tilde{X}_{RTL} - \tilde{X}_{RTL}^{(j)} \right\|_2 \right] \quad (8)$$

where the tilde denotes normalization by the per-feature maximum across the corpus; the weights 0.8 and 0.2 reflect that post-synthesis area is the stronger indicator of transfer quality.

A structural override precedes (8): finite state machines declare multiple state parameters and synthesize to a small state register with combinational next-state logic, so their gate count is far lower than their line count suggests, and designs satisfying $n_{param} \geq 3 \wedge n_{case} \geq 1$ are therefore matched to the counter class regardless of estimated area.

Configuration parameters are then initialized from the matched design's successful runs. Clock period, utilization, and density are set to the medians over pass runs, and the clock period is scaled by the ratio of

query to reference line count, clamped to [0.8, 3.0], so that larger designs receive a proportionally relaxed timing constraint. The result is projected into Θ and written as config. json.

3.6. Reinforcement learning-based parameter tuning

The tuning component implements tabular Q-learning over the state, action, and reward definitions of section 2.2, updating the action-value function after each completed flow according to the standard Q-learning rule with learning rate $\alpha = 0.1$, discount factor $\gamma = 0.9$, and ε-greedy selection at $\varepsilon = 0.05$.

Each episode applies one action to the base configuration, executes the full OpenLane flow, and updates Q from the resulting post-route metrics. Because every episode requires a complete physical design run, the agent was trained for a limited episode budget, sufficient to demonstrate closed-loop operation against real signoff metrics but below the sample budget a tabular method requires to converge; the results of section 4 therefore derive from the surrogate pipeline of sections 3.4 and 3.5 rather than from a converged policy.

3.7. Dataset generation

Five benchmark designs implemented on the SkyWater 130 nm open-source PDK (sky130A) [3] produced the training corpus: a 10-bit up/down counter, an arithmetic logic unit, a first-in first-out buffer, a multiply-accumulate unit, and a universal asynchronous receiver-transmitter. For each, the three configuration parameters were swept within the OpenLane flow [2] using Latin hypercube sampling [24], which provides stratified coverage of each parameter dimension at a fraction of the cost of a full factorial grid. Every sampled configuration was executed end to end and die area, core area, instance count, instance area, total power, setup and hold slack, utilization, and completion status were parsed from the OpenLane reports.

The corpus comprises 193 completed runs, of which 172 (89.1%) reached signoff and 21 (10.9%) failed. Table 2 gives the per-design composition and the observed metric ranges. Failures are concentrated in the MAC and UART designs; the counter, ALU, and FIFO sweeps completed successfully at every sampled configuration. This imbalance has direct consequences for the pass/fail classifier, which are analyzed in section 4.1.

Table 2. Composition of the generated dataset

Design	Total runs	Pass	Fail	Die area range (μm^2)	Total power range (W)
Counter	26	26	0	2,179–3,369	0.000174–0.000371
ALU	52	52	0	142,474–645,399	0.000085–0.000535
FIFO	39	39	0	256,149–330,355	0.000153–0.000264
MAC	50	32	18	9,540–2,060,000	0.000210–0.001600
UART	26	23	3	307,785–624,292	0.000218–0.001815
Total	193	172	21	2,179–2,060,000	0.000085–0.001815

3.8. Evaluation protocol and reproducibility

The framework was evaluated on four designs held out entirely from training: a finite state machine with shift-register functionality, an 8-bit multiplexer, a 16-to-4 priority encoder, and an 8-tap finite impulse response filter. For each, only the Verilog source was supplied; no configuration, complexity label, or target metric was provided. The generated config.json was executed through a complete OpenLane flow and the resulting signoff metrics compared against the predictions, with error reported as $|Y_{pred} - Y_{actual}| / |Y_{actual}| \times 100$.

Beyond the three swept parameters, all remaining OpenLane settings were held constant across every run to isolate their effect: FP_IO_MODE = 1, RUN_CTS = 1, RUN_ROUTING = 1, RUN_CVC = 0, RUN_ERC = 0, SYNTH_STRATEGY = "AREA 0", and MAX_FANOUT_CONSTRAINT = 10. The worst negative slack reported corresponds to the nominal typical corner (nom_tt_025C_1v80).

All experiments were conducted on an Apple M-series MacBook Pro using the Docker container ghcr.io/efabless/openlane2:2.3.10. The reproducibility package comprises the nine Verilog benchmark sources, the Latin hypercube sampling scripts together with the parameter ranges of (2), the 193-run dataset in CSV form, the feature extraction, training,

prediction, and comparison scripts, and the eleven trained model artefacts. These materials are available from the corresponding author upon reasonable request.

IV. RESULTS AND DISCUSSION

4.1. Per-design model performance

Table 3 reports cross-validated die area prediction accuracy for the random forest surrogates against linear regression and decision tree baselines. Area models are trained on all runs of each design, since a configuration that fails signoff still yields a valid floorplan area, whereas power models are restricted to runs reaching signoff, for which total power is meaningful.

The sample counts in Table 3 differ accordingly for the MAC and UART designs, which are the only two exhibiting flow failures.

Table 3. Die area and total power prediction R^2 for trained designs (5-fold cross-validation)

Design	Area N	Area LR	Area DT	Area RF	Power N	Power LR	Power DT	Power RF
Counter	26	−1.962	−0.363	−0.335	26	−7.935	−12.448	−10.985
ALU	52	+0.999	+1.000	+0.947	52	−1.339	−0.774	−0.866
FIFO	39	+0.904	+0.935	+0.965	39	−0.859	−3.135	−1.017
MAC	50	+0.999	+0.853	+0.891	32	−0.792	−0.631	−0.400
UART	26	−21.794	+0.649	+0.733	23	−213.673	−157.915	−293.164
Average	–	–	–	+0.640	–	–	–	−0.761

LR: linear regression; DT: decision tree; RF: random forest; N: number of runs.

The ALU, FIFO, and MAC designs achieve R^2 of 0.947, 0.965, and 0.891 respectively, confirming that die area is predictable from the pre-synthesis instance-count proxy for designs whose parameter sweeps produced substantial area variation (see the feature importance analysis in section 3.4). The counter yields negative R^2 for all three estimators: its 26 configurations produced die areas within the narrow band 2,179–3,369 μm^2, so the variance of the target is small and the coefficient of determination is dominated by residual noise rather than by explained signal. Averaged across the five designs the random forest attains $R^2 = 0.640$, and repeating the cross-validation with shuffled fold assignment changes this by less than 0.06 for every design with positive R^2, confirming that the reported accuracy is not an artefact of the fold partitioning.

Power prediction behaves very differently, and the result is reported in full rather than omitted, since it delimits the scope within which the framework should be applied. No design achieves positive cross-validated R^2 for power under any estimator, and the random forest average of −0.761 indicates that the models do not outperform a per-design mean predictor. The cause follows directly from the feature set of (6): within a single design the structural descriptor is constant, so the regressor observes only clock period, utilization, and density. These parameters govern floorplan geometry and cell placement, and therefore area, but they do not determine switching activity, which dominates dynamic power and depends on input signal statistics that are unobservable without simulation. Consequently, the low single-point percentage errors reported for some designs in Tables 4 and 5 reflect proximity between the queried configuration and the per-design mean rather than genuine predictive capability. Power estimates from this framework should be interpreted as order-of-magnitude indicators only; mitigation strategies are discussed in section 4.4.

The global pass/fail classifier attains 94.3% accuracy and 86.4% balanced accuracy under stratified five-fold cross-validation over the 193-run corpus, against a majority-class baseline of 89.1%. Because the corpus is imbalanced, balanced accuracy is the more meaningful figure: the classifier recovers 16 of the 21 failing configurations, giving recall 0.76 and precision 0.73 on the failure class, while retaining precision and recall of 0.97 on the passing class.

When the same evaluation is repeated with unshuffled folds, accuracy falls to 74.1%. As the corpus is stored in design order, contiguous folds are dominated by individual designs, so this figure approximates cross-design rather than within-distribution generalization.

The gap between the two protocols quantifies how much of the classifier's performance depends on having observed configurations from the same design during training, and indicates that transferring feasibility prediction to entirely unseen design classes remain substantially harder than predicting within a known class.

4.2. Trained design validation

Table 4 presents predicted versus actual metrics for the five trained designs, the surrogate being queried at the median configuration derived from each design's successful runs and compared against metrics averaged over those runs.

Table 4. Prediction vs. actual PPA comparison for trained designs

Design	Runs	Area (μm^2)	Error	Power (W)	Error	Timing P/F	DRC
Counter	27	2,692.6	2.2%	0.000236	0.9%	Correct	0
ALU	51	190,860	24.4%	0.000125	43.1%	Wrong	0
FIFO	46	295,790	71.0%	0.000196	90.0%	Correct	0
MAC	45	818,617	13.0%	0.001217	28.2%	Wrong	0
UART	36	414,162	53.5%	0.000577	1.4%	Correct	0
Average	–	–	32.8%	–	32.7%	3/5	0

The Runs column reports the number of completed flow executions in the OpenLane run directories contributing to each average, which differs from the curated dataset counts of Table 2, where duplicate or incomplete runs were filtered during corpus construction.

Area prediction achieves 2.2% error for the counter and 13.0% for the MAC, with 24.4% for the ALU, yielding an average of 32.8% across the trained set. The FIFO and UART exhibit higher area error at 71.0% and 53.5%; both were swept over comparatively wide parameter ranges, producing greater variance in realised die area and therefore a larger deviation between the median-configuration prediction and the mean across all successful runs. Power errors average 32.7%, but as established in section 4.1 these figures reflect proximity to the per-design mean rather than learned response and should not be read as evidence of predictive power.

Timing pass/fail prediction is correct for three of five trained designs. The ALU and MAC are both misclassified in the conservative direction, forecasting failure where the flow succeeded; for the ALU the reference slack was −0.106 ns against an actual worst negative slack of +5.501 ns. This direction of error is preferable in practice, since it does not admit configurations that would violate timing in silicon. All five trained designs completed with zero design rule violations.

4.3. Unseen design validation

Table 5 presents the corresponding comparison for four designs held out entirely from training. For each, only the Verilog source was supplied; the framework identified the closest trained design through (8), generated an OpenLane configuration, and predicted metrics without any prior knowledge of the design.

Table 5. Prediction vs. actual PPA comparison for unseen test designs

Design	Closest	Area (μm^2)	Error	Power (W)	Error	Timing P/F	DRC
FSM	Counter	3,097.9	20.8%	0.000218	2.9%	Correct	0
MUX8	Counter	3,097.9	26.0%	0.000218	46.1%	Correct	0
Encoder	Counter	3,097.9	18.5%	0.000218	117.9%	Correct	0
FIR	Counter	3,097.9	59.9%	0.000218	55.0%	Correct	0
Average	–	–	31.3%	–	55.5%	4/4	0

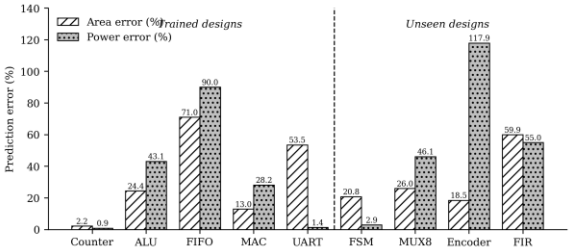

Figure 2. Area and power prediction error across all trained and unseen designs; the dashed vertical line separates trained from unseen designs

All four unseen designs achieve correct timing pass/fail classification and zero design rule violations. This is the framework's strongest validation result: from a Verilog source alone, with no design-specific tuning and no prior exposure to the design, the generated configurations produced physically manufacturable layouts in every case. Actual worst negative slack was comfortably positive throughout (+8.219 ns for the MUX8 to +12.250 ns for the FIR), and routing converged cleanly, the FIR reducing violations across successive iterations from 57 to 0 and the MUX8 from 79 to 0.

Area prediction error averages 31.3%, ranging from 18.5% for the encoder to 59.9% for the FIR. This illustrates a structural limitation of the similarity mechanism: all four unseen designs matched to the counter class and received the identical predicted area of 3,097.9 μm^2, whereas realised areas ranged from 4,189 to 7,724 μm^2. The framework correctly identified all four as small designs but could not discriminate among them, because the pre-synthesis descriptor of (4) does not capture the arithmetic replication that makes an eight-tap filter substantially larger than a multiplexer. Prediction granularity for unseen designs is therefore bounded by the granularity of the training corpus: with a single small-design class available for matching, every small design inherits the same estimate. Enlarging the corpus with additional design classes is the direct remedy, and is identified as future work in section 5.

4.4. Analysis of results and error mitigation strategies

Area prediction degrades where the query design falls between training classes, as with the FIR, or where the training sweep range was wide relative to the queried configuration, as with the FIFO and UART. Both failure modes are addressed by corpus expansion rather than by a change of estimator.

Two mitigation routes for the power limitation established in section 4.1 follow directly from its diagnosis. The first is to introduce switching activity into the feature set through simulation-based annotation, which supplies precisely the signal the current structural descriptor omits; this addresses the encoder, whose 117.9% power error is the largest observed.

The second is to adopt learned power models operating on post-placement layout representations, such as the convolutional estimator demonstrated in PowerNet [26], which have been shown to transfer across designs.

Although timing is obtained by reference lookup rather than by a learned model, as described in section 3.4, pass/fail classification was correct for seven of the nine evaluated designs. The explanation is that within the sampled parameter space the benchmark designs met timing with large positive margin, so a reference-based decision remains reliable even though the underlying slack value is not accurately predicted. This reliability should not be expected to extend to designs operating close to their timing limit, and constitutes an explicit boundary on the current framework.

Across all nine evaluated designs, every generated configuration completed with zero design rule violations. This is the most practically consequential result, since a configuration that is inaccurate in its metric estimates but reliably manufacturable remains useful for early-stage screening, whereas one that is metrically accurate but produces unroutable layouts does not.

4.5. Comparison with prior work

Table 6 compares CSF-MORL against representative prior approaches on capabilities relevant to open-source design space exploration.

Two distinctions warrant emphasis. Prior reinforcement learning approaches to physical design compute their reward from the single stage they optimize, typically placement or macro placement, whereas CSF-MORL computes (3) from post-route signoff metrics extracted after the complete flow, so that routing outcomes govern parameters acting at synthesis and floorplanning.

Table 6. Comparison with prior work

Method	Learning-based	Multi-objective	Post-route feedback	Open-source
Mirhoseini et al. [12]	Yes	No	No	No
RL parameter tuning [14]	Yes	Partial	No	No
ML timing prediction [4]	Yes	No	No	Partial
DRC hotspot detection [17]	Yes	No	No	No
CSF-MORL (ours)	Yes	Yes	Yes	Yes

Separately, the framework operates entirely within an open-source toolchain on an open process design kit and requires no commercial EDA licence, which is a precondition for the small-team, academic, and rapid-prototyping use cases that motivate this work.

4.6. Complexity, deployment implications, and limitations

Inference cost is dominated by random forest evaluation, which is O (N_t log n) per predicted metric for N_t = 300 trees, amounting to microseconds on a modern CPU. The complete pipeline completes in under two seconds per design on a standard laptop, against 8 to 45 minutes for a single OpenLane execution, and the eleven trained artefacts together occupy approximately 5 MB, so the framework requires no GPU. For an exploration comprising 50 candidate configurations, replacing full flow invocations with surrogate evaluation reduces exploration time by approximately three orders of magnitude.

The practical value of the framework therefore lies in feasibility screening rather than in signoff-quality estimation. For an embedded or internet of things development team, a designer can obtain a manufacturable OpenLane configuration and an order-of-magnitude area estimate for an arbitrary Verilog module in seconds, enabling rapid architectural comparison before committing compute to full flows. Because only the Verilog source is required as input, the framework interoperates naturally with heterogeneous RTL from different teams or generators, and its small footprint permits integration into cloud-based collaborative design platforms.

Two scalability limits must be acknowledged. The surrogates are calibrated to the SkyWater 130 nm PDK, and advanced nodes introduce multiple patterning and pin-access constraints that dominate routability below 14 nm [22] and are not observable from RTL-level features, so node-specific retraining would be required; and for large hierarchical designs the per-design approach would require block-level composition, with the parameter space of (2) expanding to include macro placement and power-grid parameters. The framework principles transfer directly to commercial flows such as Cadence Innovus and Synopsys IC Compiler, where the same reward could be computed from tool-reported quality-of-results metrics; calibration against silicon measurements and validation on larger benchmark suites [23] remain necessary before industrial deployment.

The study has four further limitations. The corpus of 193 runs across five design classes is modest, with its 21 failures concentrated in two designs, limiting the diversity of infeasible configurations available to the classifier. All four unseen designs matched to the counter class, so transfer to structurally distinct targets is not demonstrated. Power prediction does not exceed mean-predictor performance, timing is obtained by reference lookup, and congestion and wirelength are not modelled at all. Finally, the reinforcement learning component was trained for a limited episode budget, sufficient to demonstrate closed-loop operation but below the sample budget a tabular method requires to converge, so the reported results derive from the surrogate pipeline rather than a converged policy. These define the future work agenda of section 5 rather than undermining the principal claim: that a lightweight surrogate trained on a modest corpus can generate physically manufacturable configurations for previously unseen RTL at a small fraction of the cost of exhaustive flow-based exploration. The influence of the reward weights (w_1 = 0.4, w_2 = w_3 = 0.3), fixed for all reported runs, on the configuration selected by the framework is examined in section 4.7.

4.7. Sensitivity to reward weighting

Because the reward of (3) is a weighted sum of already-measured signoff metrics, the influence of the weighting coefficients on the selected configuration can be assessed directly on the executed corpus, without additional OpenLane runs. For each design, the feasible (signoff-reaching) configurations are

ranked by the cost J(θ) of (1) under a given weighting, and the minimizing configuration is recorded. The weight simplex $w_1 + w_2 + w_3 = 1$ was swept at a resolution of 0.05, giving 231 distinct weight vectors that span area-, power-, and slack-dominant regimes.

Table 7 reports, for each design, the number of distinct optimal configurations selected across the simplex and the fraction of weight vectors that reproduce the baseline selection at $(w_1, w_2, w_3) = (0.4, 0.3, 0.3)$. Under the raw-magnitude reward of (3), configuration selection is highly stable: between one and four distinct configurations are selected across the entire simplex, and the baseline configuration is reproduced for 82.3% to 100% of weight vectors. This follows directly from the scaling in (3). Within a single design the die area term (order 10^3–10^6 μm^2) exceeds the scaled power term $w_2 \cdot 10^3 \cdot P$ (order 10^{-1}) by several orders of magnitude, and the scaled slack term is zero for the feasible configurations of every design except the UART. The raw reward is therefore area-dominated, and the minimizing configuration coincides with the minimum-area feasible configuration for all five designs, independent of the weighting.

Table 7. Sensitivity of the selected configuration to reward weighting across the 231-point weight simplex

Design	Feasible configs	Raw reward, Eq. (3): Distinct optima	Raw reward, Eq. (3): Baseline share	Normalized reward: Distinct optima	Normalized reward: Baseline share
Counter	26	3	90.9%	5	21.6%
ALU	52	3	90.9%	3	90.9%
FIFO	39	4	82.3%	6	64.9%
MAC	32	2	99.6%	2	99.6%
UART	23	1	100.0%	1	100.0%

To expose the trade-off masked by this magnitude imbalance, the analysis was repeated with each objective min–max normalized to [0, 1] within the feasible set of each design, so that the weights express relative rather than absolute priority. Sensitivity increases for the two designs whose objectives are not jointly minimized by a single configuration the counter (five distinct optima; baseline share 21.6%) and the FIFO (six; 64.9%) whereas the ALU (90.9%), MAC (99.6%), and UART (100%) remain stable because a single configuration simultaneously minimizes their competing objectives. Figure 3 summarizes both analyses.

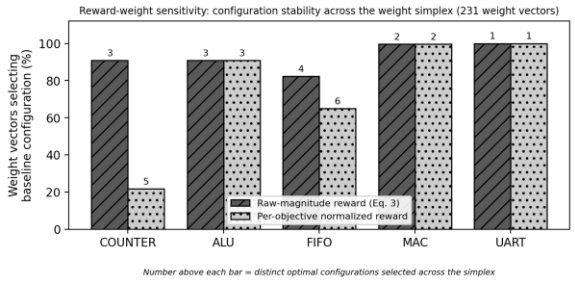

Figure 3. Reward-weight sensitivity: fraction of the 231-point weight simplex that reproduces the baseline configuration, under the raw-magnitude reward of (3) and under a per-objective normalized reward. Numbers above bars give the count of distinct optimal configurations selected across the simplex.

Two conclusions follow. First, the framework's configuration selection is robust to the precise choice of reward weights, which justifies reporting results at a single fixed weighting rather than tuning it per design. Second, the area dominance of the raw-magnitude reward means that the multi-objective character of (3) is fully expressed only once the objectives are placed on a common scale; per-objective normalization is therefore recommended when power or slack is intended to influence selection materially, and is adopted as a refinement in the future work of section 5.

V. CONCLUSION

This paper introduced CSF-MORL, a cross-stage feedback-driven learning framework for early-stage PPA estimation and configuration generation in OpenLane ASIC physical design. The framework combines per-design random forest models, a global pass/fail classifier, an area-aware design similarity

measure, and a feedback-driven optimization loop to automate early-stage physical design parameter selection on the SkyWater 130 nm open-source PDK. Experimental validation on five training designs (193 runs) and four unseen designs shows zero DRC violations for all nine designs, 31.3% average area prediction error on unseen designs, correct timing pass/fail classification on all unseen designs, and a reduction in exploration time of approximately three orders of magnitude, with each prediction completing in under two seconds. In terms of reconfigurable design space exploration principles, the framework demonstrates that a single surrogate model, retargetable across heterogeneous user-defined RTL inputs without retraining, can replace repeated full-flow executions during early exploration, thereby lowering the computational barrier to custom silicon for embedded and IoT system designers.

Future work will proceed in several directions: i) adding placement-aware features (floorplan aspect ratio, pin density, anticipated routing demand) collected after the floorplan stage to enable wirelength and congestion prediction, which the current corpus does not support; ii) incorporating simulation-based switching activity annotation and transferable learned power models [26] for switching-dominated designs such as encoders and FIR filters; iii) enlarging the training corpus with pipelined processors, memories, and mixed-signal interfaces; iv) replacing tabular regressors with graph neural networks operating on synthesized netlist graphs [19] and extending predictive modeling to the CTS stage [27]; v) adopting per-objective normalized rewards and learned weightings to strengthen the multi-objective trade-off identified in section 4.7; vi) extensions targeted at reconfigurable, embedded, and IoT ecosystems, including FPGA-based acceleration, adaptive parameter tuning for edge deployment, integration with cloud-based design automation platforms, validation on commercial EDA flows and advanced nodes, and federated learning across organizations.

ACKNOWLEDGMENTS

The authors thank the Department of Electronics and Communication Engineering, CHH. Shahu College of Engineering, Chhatrapati Sambhajinagar, for providing the research facilities and computational resources used in this study.

FUNDING INFORMATION

Authors state no funding involved.

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data that support the findings of this study, including the OpenLane run dataset, the feature extraction and prediction scripts, and the trained model artefacts, are available from the corresponding author, SGJ, upon reasonable request.

REFERENCES

[1] B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, 2nd ed. Cham, Switzerland: Springer, 2022, doi: 10.1007/978-3-030-96415-3.

[2] M. Shalan and T. Edwards, “Building OpenLANE: A 130nm OpenROAD-based tapeout-proven flow,” in *Proc. IEEE/ACM Int. Conf. Comput. -Aided Design (ICCAD)*, San Diego, CA, USA, 2020, pp. 1–6, doi: 10.1145/3400302.3415735.

[3] T. Ansell and R. Wilton, “Democratising design: The SkyWater open-source PDK,” in *Proc. IEEE Hot Chips Symp.*, Cupertino, CA, USA, 2020, pp. 1–18.

[4] E. C. Barboza, N. Shukla, Y. Chen, and J. Hu, “Machine learning-based pre-routing timing prediction with reduced pessimism,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, Las Vegas, NV, USA, 2019, pp. 1–6, doi: 10.1145/3316781.3317857.

[5] G. Huang *et al*., “Machine learning for electronic design automation: A survey,” *ACM Trans. Design Autom. Electron. Syst.*, vol. 26, no. 5, pp. 1–46, Sep. 2021, doi: 10.1145/3451179.

[6] Y. Zhou, H. Ren, Y. Zhang, B. Keller, B. Khailany, and Z. Zhang, “PRIMAL: Power inference using machine learning,” in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, Las Vegas, NV, USA, 2019, pp. 1–6, doi: 10.1145/3316781.3317884.

[7] Y. Lin, Z. Jiang, J. Gu, W. Li, S. Dhar, H. Ren, B. Khailany, and D. Z. Pan, "DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 4, pp. 748–761, Apr. 2021, doi: 10.1109/TCAD.2020.3003843.

[8] A. Agnesina, K. Chang, and S. K. Lim, "VLSI placement parameter optimization using deep reinforcement learning," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Virtual Event, USA, 2020, pp. 1–9, doi: 10.1145/3400302.3415690.

[9] J. Jung, A. B. Kahng, S. Kim, and R. Varadarajan, "METRICS2.1 and flow tuning in the IEEE CEDA robust design flow and OpenROAD," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Munich, Germany, 2021, pp. 1–9, doi: 10.1109/ICCAD51958.2021.9643541.

[10] W. L. Neto, M. Austin, S. Temple, L. Amaru, X. Tang, and P.-E. Gaillardon, "LSOracle: A logic synthesis framework driven by artificial intelligence," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Westminster, CO, USA, 2019, pp. 1–6, doi: 10.1109/ICCAD45719.2019.8942145.

[11] T. Ajayi et al., "Toward an open-source digital flow: First learnings from the OpenROAD project," in Proc. ACM/IEEE Design Autom. Conf. (DAC), Las Vegas, NV, USA, 2019, pp. 1–4, doi: 10.1145/3316781.3326334.

[12] A. Mirhoseini et al., "A graph placement methodology for fast chip design," Nature, vol. 594, no. 7862, pp. 207–212, Jun. 2021, doi: 10.1038/s41586-021-03544-w.

[13] A. B. Kahng, "Machine learning applications in physical design: Recent results and directions," in Proc. ACM Int. Symp. Physical Design (ISPD), Monterey, CA, USA, 2018, pp. 68–73, doi: 10.1145/3177540.3177554.

[14] A. Agnesina, S. Pentapati, and S. K. Lim, "A general framework for VLSI tool parameter optimization with deep reinforcement learning," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS) Workshop on Machine Learning for Systems, Virtual Event, 2020, pp. 1–7.

[15] A. Agnesina, P. Rajvanshi, T. Yang, G. Pradipta, A. Jiao, B. Keller, B. Khailany, and H. Ren, "AutoDMP: Automated DREAMPlace-based macro placement," in Proc. ACM Int. Symp. Physical Design (ISPD), Virtual Event, USA, 2023, pp. 149–157, doi: 10.1145/3569052.3578923.

[16] A. B. Kahng, R. Varadarajan, and Z. Wang, "RTL-MP: Toward practical, human-quality chip planning and macro placement," in Proc. ACM Int. Symp. Physical Design (ISPD), Virtual Event, Canada, 2022, pp. 3–11, doi: 10.1145/3505170.3506731.

[17] R. Liang, H. Xiang, D. Pandey, L. Reddy, S. Ramji, G.-J. Nam, and J. Hu, "DRC hotspot prediction at sub-10nm process nodes using customized convolutional network," in Proc. ACM Int. Symp. Physical Design (ISPD), Taipei, Taiwan, 2020, pp. 135–142.

[18] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), San Diego, CA, USA, 2018, pp. 1–8.

[19] Y.-C. Lu, S. Pentapati, and S. K. Lim, "VLSI placement optimization using graph neural networks," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS) Workshop on Machine Learning for Systems, Virtual Event, 2020, pp. 6–12.

[20] M. B. Alawieh, W. Li, Y. Lin, L. Singhal, M. A. Iyer, and D. Z. Pan, "High-definition routing congestion prediction for large-scale FPGAs," in Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC), Beijing, China, 2020, pp. 26–31.

[21] A. F. Tabrizi, L. Rakai, N. K. Darav, I. Bustany, L. Behjat, S. Xu, and A. Kennings, "A machine learning framework to identify detailed routing short violations from a placed netlist," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2018, pp. 1–6, doi: 10.1109/DAC.2018.8465835.

[22] W.-T. J. Chan, P.-H. Ho, A. B. Kahng, and P. Saxena, "Routability optimization for industrial designs at sub-14nm process nodes using machine learning," in *Proc. ACM Int. Symp. Physical Design (ISPD)*, Portland, OR, USA, 2017, pp. 15–21, doi: 10.1145/3036669.3036681.

[23] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21,

no. 2, pp. 239–245, May 1979, doi: 10.1080/00401706.1979.10489755.

[24] M.-C. Kim, J. Hu, J. Li, and N. Viswanathan, "ICCAD-2015 CAD contest in incremental timing-driven placement and benchmark suite," in *Proc. IEEE/ACM Int. Conf. Comput. -Aided Design (ICCAD)*, Austin, TX, USA, 2015, pp. 921–926.

[25] F. Pedregosa *et al*., "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

[26] Z. Xie *et al*., "PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network," in *Proc. Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Beijing, China, 2020, pp. 13–18, doi: 10.1109/ASP-DAC47756.2020.9045574.

[27] Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Westminster, CO, USA, 2019, pp. 1–8, doi: 10.1109/ICCAD45719.2019.8942063.