ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Machine-Learning-Based PPA Optimization for VLSI Design-Space Exploration

Pushkar Raj, Partap Singh

Department of Computer Science and Engineering, Quantum University, Roorkee, India

Published November 2026

Abstract

Power, performance, and area (PPA) optimization in digital VLSI design requires repeated evaluation of parameterized circuits under synthesis, timing, and implementation constraints. The cost of these evaluations can make broad design-space exploration impractical, particularly when architecture, RTL structure, technology context, and physical-design settings interact nonlinearly. This paper specifies an implementation-aware machine-learning framework that combines parameterized Verilog generation, functional verification, synthesis, physical implementation, feature extraction, surrogate prediction, and NSGA-II-based multi-objective candidate selection. The proposed benchmark scope includes adder architectures, multiplexers, counters, comparators, and arithmetic logic units. Random Forest, XGBoost, and artificial neural network regressors are considered as candidate surrogate models. The framework explicitly separates pre-implementation inputs, post-implementation measurements, model predictions, and verification status. Only functionally verified candidates with traceable implementation reports may contribute measured labels to the authoritative dataset or the final measured Pareto set. ASIC and FPGA experiments are treated as separate tracks because their area, timing, power, and resource metrics are technology-and flow-dependent. A reproducible experimental protocol is defined for model comparison, cross-family generalization, matched-budget search comparison, ablation, and robustness analysis. No measured PPA values, model scores, or optimization improvements are claimed in this manuscript; those results must be populated after the experiments are executed.

Keywords: active learning; artificial neural network; machine learning; NSGA-II; OpenROAD; random forest; surrogate modeling; XGBoost; Yosys

1 Introduction

Digital integrated circuits are evaluated against several competing objectives. A design with lower cell area may have a longer critical path, while a faster implementation may require additional logic, buffering, routing resources, or power. The relationship between RTL-level structure and final implementation metrics is not generally monotonic: synthesis transformations, technology mapping, placement, routing, clock constraints, and the selected process or FPGA device can change the ranking of candidate architectures.

Design-space exploration (DSE) becomes challenging when the design is parameterized by circuit family, architecture, operand width, block size, pipeline options, synthesis directives, clock targets, and physical-design settings. Exhaustive evaluation may be too expensive, while manual selection can be difficult to reproduce and may not generalize to unseen configurations. A surrogate model can estimate implementation objectives for unevaluated candidates, allowing an optimizer to prioritize a smaller set of expensive evaluations.

This work proposes a closed-loop workflow in which machine-learning predictions guide candidate selection, but do not replace functional verification or implementation measurement. The framework connects RTL generation, simulation or formal checking, synthesis, physical implementation, feature extraction, surrogate fitting, multiobjective optimization, and measured feedback. The central research question is whether this loop can reduce implementation evaluations while preserving useful measured Pareto-front quality.

1.1 Scope and contributions

The initial benchmark families are: (i) ripple-carry, carry-lookahead, carry-select, Kogge–Stone, Brent–Kung, and hybrid-prefix adders; (ii) multiplexers; (iii) counters; (iv) comparators; and (v) arithmetic logic units (ALUs). The first implementation phase should prioritize a coherent, well-verified subset rather than claiming full coverage before all generators and flows exist.

- An implementation-aware pipeline from parameterized RTL to verified synthesis and physical-implementation observations.
- A feature and provenance schema that separates configuration inputs, pre-implementation features, post-implementation measurements, predictions, and verification outcomes.
- A comparison of Random Forest, XGBoost, and ANN surrogate models using leakage-aware validation.
- An NSGA-II candidate-selection loop with measured feedback and a final Pareto set formed only from verified measurements.
- A reproducible experiment plan covering prediction quality, generalization, search baselines, ablations, and robustness.

2 Background and related work

Fast adder architectures provide a useful benchmark because they expose different trade-offs between logic depth, wiring, fan-out, and area. Ripple-carry adders have a serial carry dependency. Carry-lookahead structures compute group generate/propagate terms to reduce carry dependence. Carry-select structures compute alternative carry cases and select between them, typically trading additional hardware for reduced carry-path delay. Parallel-prefix adders use associative prefix computation; their topology affects depth, fan-out, wiring, and physical implementation. These architectural tendencies motivate experimentation, but they do not establish a universal PPA ranking: the ranking must be measured under a specified library, device, constraints, and implementation flow.

Surrogate modeling is widely used when direct objective evaluation is expensive. Tree ensembles can model nonlinear interactions in mixed feature spaces, while neural networks can approximate nonlinear mappings when adequate data and regularization are available. Prediction quality alone is insufficient for DSE: the model must rank candidates usefully near feasibility boundaries and preserve promising regions of the Pareto front.

NSGA-II uses non-dominated sorting, elitism, and a crowding-distance mechanism to maintain a diverse set of solutions during multi-objective search [1]. In this framework, NSGA-II operates on predicted objectives during candidate generation. The selected candidates are then functionally checked and physically evaluated. This distinction prevents the predicted Pareto set from being misreported as the measured Pareto set.

3 Research questions and hypotheses

The research questions and the evidence required to answer them are listed in Table 1.

Table 1. Research questions and required evidence.

ID	Research question	Evidence required
RQ1	Can circuit and flow features predict measured area, delay, and power with useful held-out accuracy?	Held-out MAE, RMSE, R^2, calibration/uncertainty, and ranking metrics.
RQ2	Can ML-guided active search reduce expensive implementation evaluations compared with matched baselines?	Equal-budget comparisons using verified measurements, hypervolume, coverage, and evaluation count.
RQ3	How well do models generalize across circuit families, widths, and implementation conditions?	Group-based and leave-one-family/width-out evaluation.
RQ4	Which feature groups and feedback mechanisms contribute to prediction and search performance?	Controlled ablations with fixed splits and evaluation budgets.

H1 (prediction hypothesis): circuit and implementation-context features contain information that can predict selected measured PPA targets on held-out configurations. H2 (search hypothesis): active measured feedback can improve the quality of the measured feasible Pareto set per expensive implementation evaluation relative to at least one matched baseline. H3 (transfer hypothesis): prediction performance will vary across families, widths, and technology/flow conditions; transfer must therefore be measured rather than assumed. These are hypotheses, not findings.

4 Problem formulation

Let x denote a candidate configuration. It contains the circuit family, architecture, RTL parameters, operand width, permitted synthesis options, implementation settings, technology or FPGA-device context, and design constraints. Let A(x) denote implementation area, D(x) the reported critical-path delay, and P(x) the power estimate or measurement produced by a specified flow. The primary three-objective formulation is:

$$\min f(x) = [A(x),\ D(x),\ P(x)] \tag{1}$$

subject to functional validity V(x) = 1 and implementation constraints g_j(x) ≤ 0 for j = 1, ..., J. Examples include maximum area, maximum delay, maximum power, minimum slack, maximum resource utilization, or zero permitted timing violations. The numerical limits must be selected before the evaluation campaign and recorded with the configuration. A candidate with an unverified functional result is not feasible for the final Pareto set.

4.1 Energy and power-delay product

Power-delay product (PDP) is defined as P(x)·D(x). With power in mW and delay in ns, the product has units of pJ. PDP is a derived figure of merit and must not be labelled as energy per operation without a corresponding operational definition. Energy per operation requires a defined operation, activity pattern, measurement or estimation interval, and energy accounting boundary. If energy is reported as a separate objective, the manuscript must define how it is obtained and explain why it is not redundant with the selected power and delay objectives. To avoid redundant optimization, the default protocol uses area, delay, and power as primary objectives and reports PDP as a derived

metric; energy is optional and only included when measured or estimated under a defensible, documented methodology.

4.2 ASIC and FPGA tracks

ASIC and FPGA results shall be evaluated and reported in separate tracks. ASIC area may refer to standard-cell area or another explicitly defined implementation area; FPGA results may include LUTs, flip-flops, DSP blocks, BRAM, and device-specific utilization. These quantities are not interchangeable. Delay, power, and feasibility are also conditioned on different libraries, devices, timing models, and implementation tools. A combined cross-platform model is permitted only if platform identity is explicitly encoded and the study validates transfer; raw ASIC and FPGA values must not be pooled as though they share a common measurement scale.

5 Proposed framework

The workflow consists of nine logical stages: configuration generation; RTL analysis and feature extraction; functional verification; synthesis; physical implementation; verified dataset construction; surrogate fitting and prediction; NSGA-II candidate selection; and final validation with measured feedback. Each record is assigned a unique configuration identifier and a provenance manifest. The overall flow is shown in Figure 1.

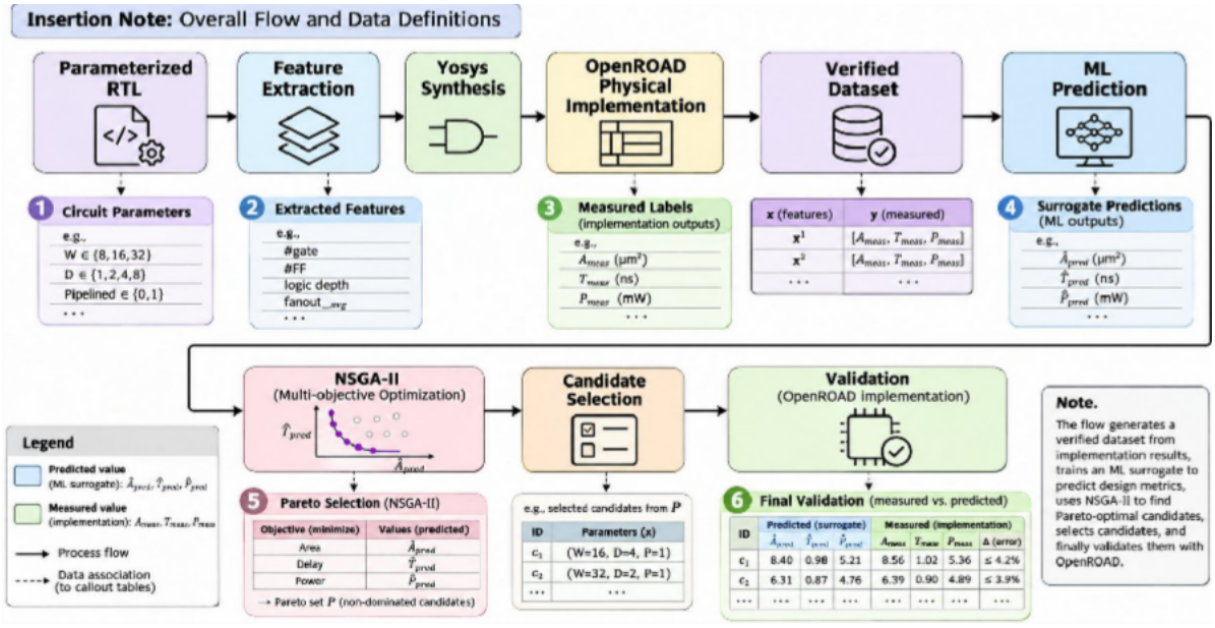


Figure 1. Conceptual flow from parameterized RTL through feature extraction, synthesis, physical implementation, dataset construction, prediction, optimization and validation.

5.1 RTL generation and functional verification

Each generator shall produce parameterized RTL from a validated configuration schema. The schema must reject illegal parameter combinations before simulation. The verification environment shall compare the generated design against a trusted reference model. Directed tests shall cover reset behavior (for sequential designs), minimum and maximum operands, carry/borrow boundaries, all control combinations, signedness, overflow semantics, and architecture-specific corner cases. Randomized tests shall use recorded seeds. For small widths, exhaustive input enumeration is preferred. Formal equivalence or property checking should be used where practical and supported.

Simulation pass status must not be inferred from a small number of random vectors alone. Each record shall include test count, directed/exhaustive coverage description, random seed, reference-model version, simulator version, log path, failed vectors, and final status. A failure blocks the candidate from implementation unless the experiment explicitly studies failed candidates; failed candidates must never be labelled functionally feasible.

5.2 Synthesis and physical implementation

Yosys is proposed for open-source RTL synthesis and netlist/statistics extraction. The synthesis flow shall record the Yosys version, script hash, technology mapping files, library corner, constraints, warnings, and generated reports. OpenROAD may be used for ASIC-oriented floorplanning, placement, clock-tree synthesis, routing, timing analysis, and report generation when a supported platform and technology files are available. The exact stages executed must be stated; a generic synthesis result is not a substitute for a completed physical implementation.

AMD Vivado may be used for a separate FPGA track. The exact FPGA part, board where applicable, speed grade, Vivado version, constraints, and synthesis/implementation options must be recorded. FPGA resource metrics and ASIC standard-cell area must be reported separately.

5.3 Verified dataset construction

A dataset row represents one unique circuit configuration evaluated under one complete set of flow and technology conditions. A repeated run with a different seed or implementation condition is a separate observation linked to the same logical configuration. The authoritative dataset stores measured targets only when the corresponding report exists and the record passes the required validation gates. Predicted values are stored in a separate prediction table and never overwrite measured labels. Figure 2 shows the feature and provenance architecture.

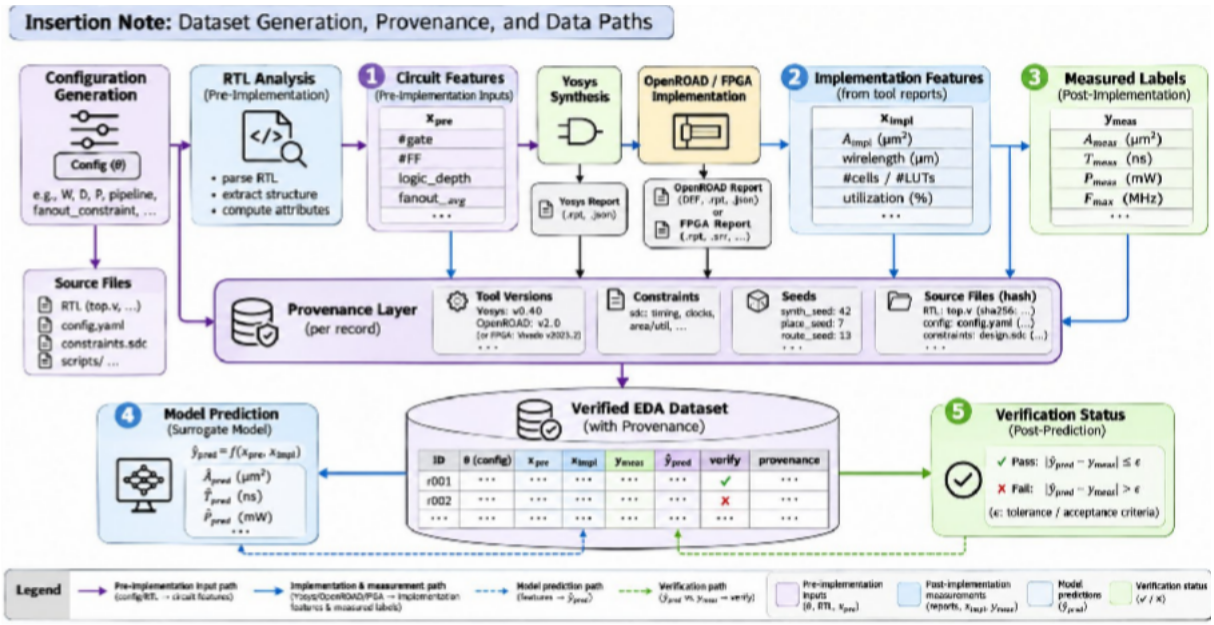


Figure 2. Conceptual feature and provenance architecture.

5.4 Surrogate prediction and active search

The surrogate model estimates one or more implementation objectives for candidates that have not yet been implemented. The model may also estimate feasibility or uncertainty, but these estimates

are screening signals only. NSGA-II ranks candidates using predicted objective vectors and constraint estimates, maintains diversity, and proposes a batch for evaluation. The selected batch is then functionally verified, synthesized, and physically implemented. Actual results are appended to the measured dataset, the model is retrained according to the predefined update schedule, and the search continues until the stopping rule is met. The closed-loop process is shown in Figure 3.

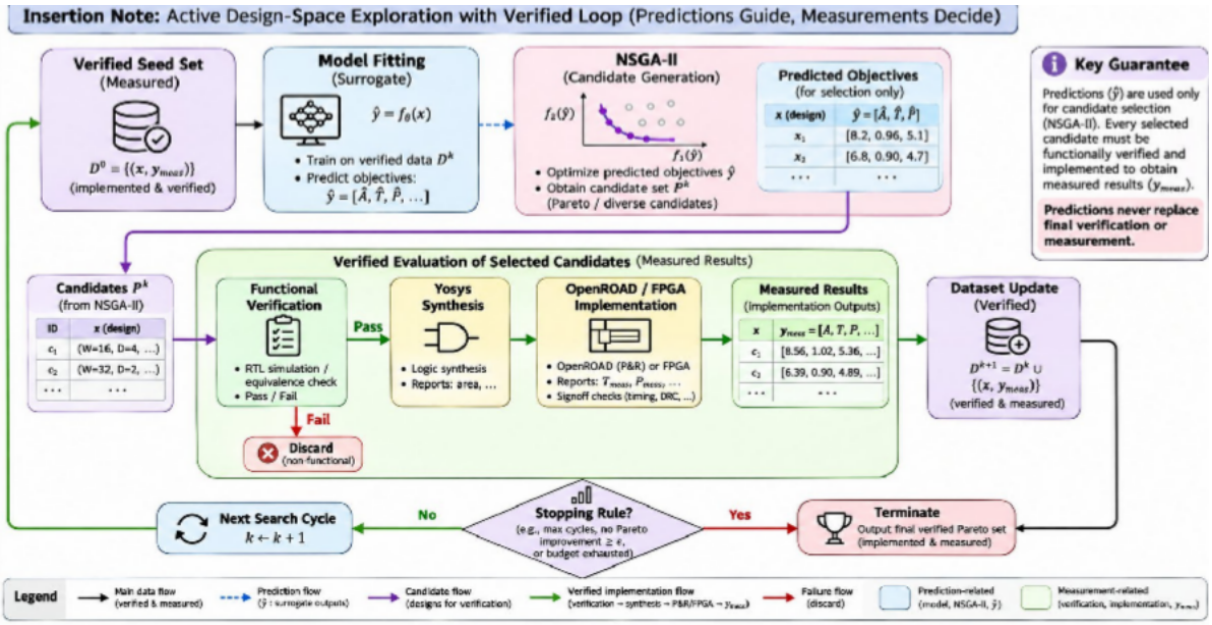


Figure 3. Conceptual closed-loop active evaluation process. Predicted objectives guide candidate selection; only verified implementation measurements update the authoritative dataset and measured Pareto front.

6 Feature engineering and data governance

Feature engineering must respect the time at which a feature becomes available. A model intended to predict implementation results before physical design cannot use post-route congestion, final timing, or final utilization as input features. Such variables would leak information from the target-producing stage into the predictor. Two prediction tasks may be defined: (i) early prediction using configuration, RTL, and synthesis-stage features; and (ii) late-stage prediction using additional intermediate implementation features. The feature set and intended prediction point must be fixed before model evaluation.

The feature groups and their availability are summarized in Table 2.

Table 2. Feature groups, availability stage and provenance rules.

Feature group	Examples	Availability stage	Provenance / rule
Configuration identity	Family, architecture, topology, operand width, signedness, parameter values	Before implementation	Generator manifest; categorical encoding and schema version
RTL structure	Hierarchy depth, operator counts, estimated logic depth, fan-out proxies, mux structure	RTL analysis	Parser/static estimator; extraction script version
Synthesis	Mapped cell counts, cell-area estimate, logic depth, warnings, synthesis runtime	After synthesis	Yosys or FPGA synthesis reports; report parser version

Feature group	Examples	Availability stage	Provenance / rule
Physical implementation	Placement utilization, congestion metrics, routed wire indicators, slack, post-route timing	After relevant implementation stage	OpenROAD or Vivado reports; stage and units recorded
Technology / flow context	Library/device, process corner, voltage, temperature, clock constraints, tool versions	Before or during run	Versioned flow configuration
Measured targets	Area, delay, power, resource counts, timing violations; energy only if defined	After measurement	Raw reports, measurement method, activity assumptions
Verification / provenance	Functional status, test coverage, seed, source hashes, tool versions, report paths	Throughout	Immutable manifest and log references

Every feature must have a name, definition, unit, data type, valid range, extraction stage, missing-value policy, and provenance source. Categorical variables such as architecture and technology must be encoded without implying an artificial ordinal relationship. Missingness must be represented explicitly where it has semantic meaning. Features unavailable at inference time must be excluded from that prediction task.

6.1 Leakage-aware splitting

Random row-level splitting can overestimate generalization when near-identical configurations, repeated seeds, or related parameterizations appear in both training and test sets. The primary evaluation shall therefore use grouped splits. All records belonging to the same logical configuration family or near-duplicate configuration group must be assigned to a single split. Additional transfer experiments shall use leave-one-width-out and leave-one-family-out partitions. Preprocessing, feature selection, and hyperparameter tuning must be fitted using training data only; the test set is reserved for final evaluation.

6.2 Dataset record schema

The fields of each dataset record are defined in Table 3.

Table 3. Dataset record schema.

Field	Example / required content	Rule
record_id	Unique immutable identifier	Required
configuration	Family, architecture, width, RTL parameters	Required
flow_id	ASIC or FPGA flow and complete settings	Required
source_hash	RTL and generator source hashes	Required
verification	Pass/fail, tests, seed, log reference	Required before measured-feasible label
features_pre	Configuration and RTL features	Only features available before target stage
features_post	Synthesis / physical features	Stage-specific prediction task only
measured_targets	Area, delay, power, resource data and units	Must link to raw report

Field	Example / required content	Rule
prediction_record	Model ID, dataset version, predictions, uncertainty	Stored separately from measured targets
feasibility	Constraint values and pass/fail results	Computed from verified measurements
pareto_status	Measured non-dominated status within defined comparison set	Recomputed after dataset updates
provenance	Tool versions, scripts, libraries, constraints, seeds, timestamps	Required

7 Predictive modeling

Three model families are proposed. Random Forest regression provides a nonlinear tree-ensemble baseline and can capture interactions without requiring feature scaling for the tree model itself. XGBoost provides gradient-boosted decision trees and may capture complex interactions efficiently. An ANN provides a flexible nonlinear model but generally requires careful scaling, regularization, architecture selection, and adequate training coverage. These are candidate models; no model is assumed to be best in advance.

7.1 Training protocol

1. Define target variables, units, prediction stage, and feature availability before training.
2. Use grouped training, validation, and test partitions; perform hyperparameter selection on training/validation data only.
3. Fit imputation, scaling, categorical encoding, and feature selection within each training fold to prevent leakage.
4. Compare models on identical splits and target definitions.
5. Report per-target MAE, RMSE, R^2, and ranking performance; include confidence intervals or uncertainty calibration where supported.
6. Retain a simple baseline (for example, training-set mean or a linear model) to determine whether complex models add predictive value.

7.2 Evaluation metrics

$$\mathrm{MAE} = \frac{1}{n}\sum_i \left|y_i - \hat{y}_i\right| \tag{2}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_i \left(y_i - \hat{y}_i\right)^2} \tag{3}$$

$$R^2 = 1 - \frac{\sum_i \left(y_i - \hat{y}_i\right)^2}{\sum_i \left(y_i - \bar{y}\right)^2} \tag{4}$$

MAE and RMSE must be reported in the target's physical units. R^2 is dataset-dependent and can be negative; it should not be interpreted as a universal measure across datasets with different target variance. If target magnitudes differ substantially, normalized errors may be added, but their

normalization denominator must be stated. Ranking metrics should assess whether the model orders candidates consistently with measured objectives, especially near feasibility boundaries.

7.3 Uncertainty

No fixed uncertainty percentage shall be assumed. Uncertainty may be estimated using an ensemble of independently trained models, quantile regression, bootstrap resampling, or conformal prediction, provided the method is implemented and calibrated on held-out data. The paper shall report empirical interval coverage and interval width. If uncertainty is not estimated, the paper must state that predictions are point estimates and must not display an unsupported confidence interval. Templates for reporting prediction quality are shown in Figure 4 and Figure 5.

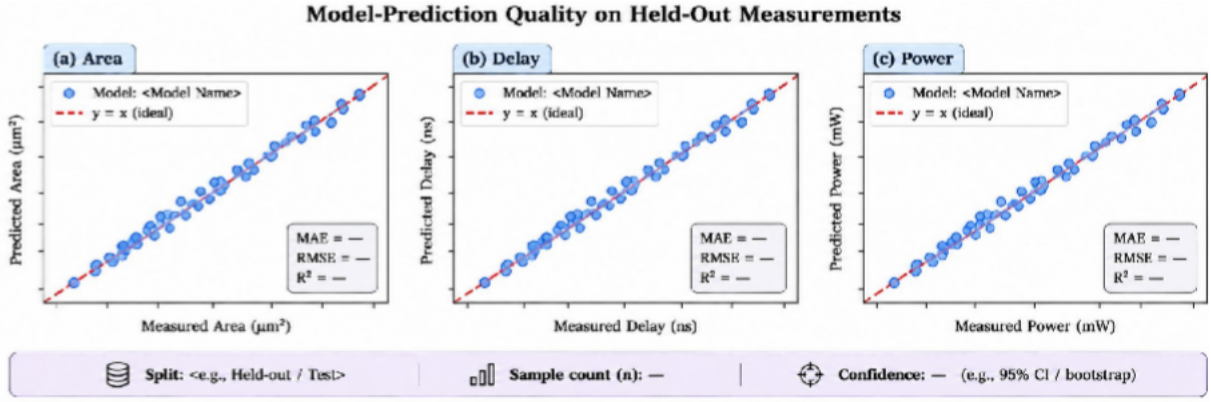


Figure 4. Model-prediction quality plot template for area, delay and power.

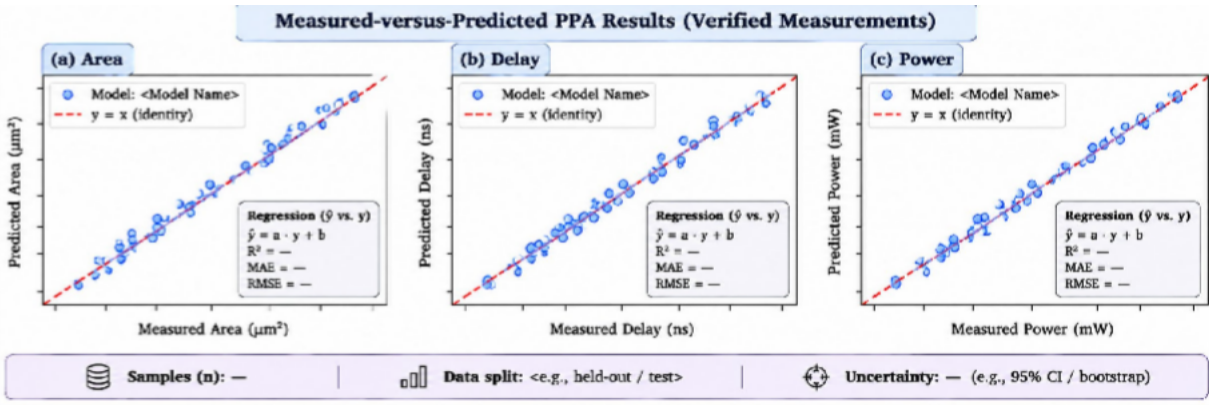


Figure 5. Measured-versus-predicted scatter-plot template.

8 NSGA-II and closed-loop optimization

NSGA-II is used to search a mixed design space containing categorical architecture choices, integer parameters such as width or block size, and continuous flow settings where permitted. Candidate encoding must enforce legal combinations. Invalid configurations should be rejected before evaluation rather than silently repaired into undocumented alternatives.

At each iteration, the surrogate predicts the objective vector for candidate configurations. The optimizer applies non-dominated sorting and diversity preservation to select a batch for expensive evaluation. Predicted feasibility can be used for screening, but the protocol should reserve some evaluation capacity for exploration, including candidates with uncertainty or underrepresented

feature values. This reduces the risk of excluding useful candidates because of model bias. Figure 6 illustrates the optimization design space.

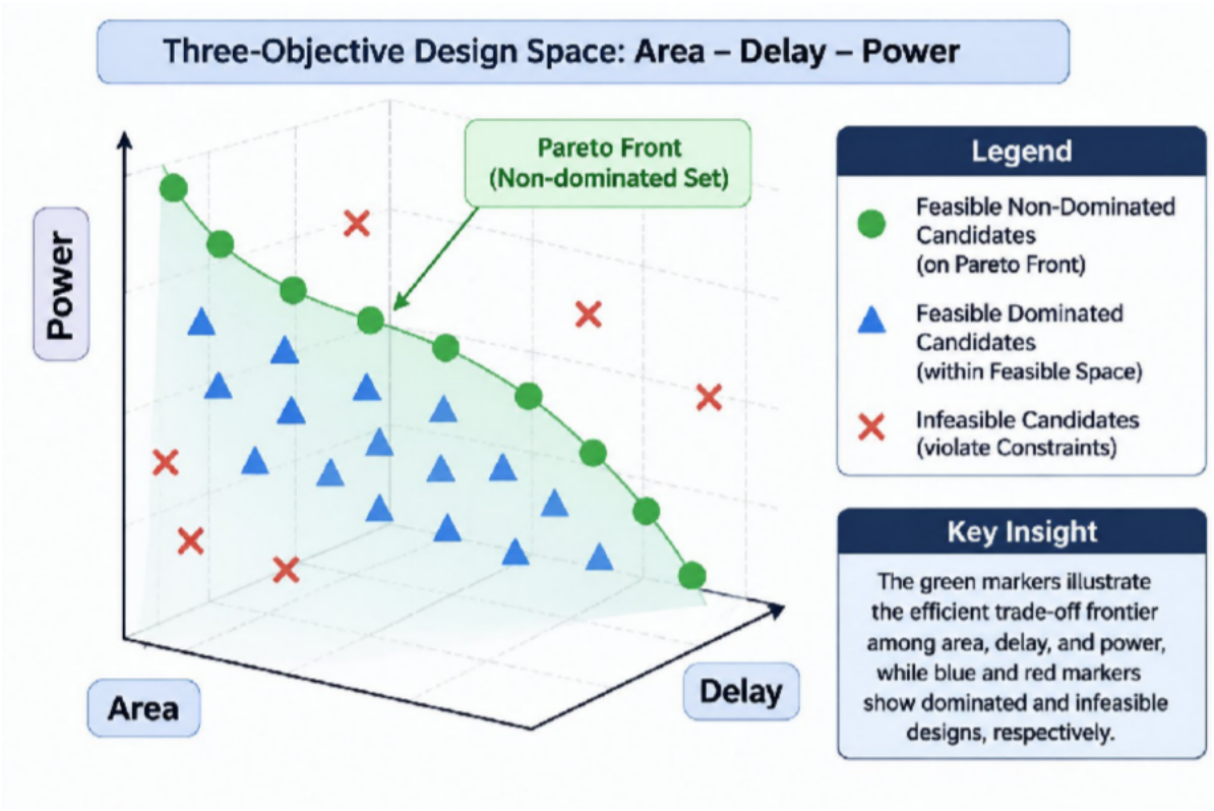


Figure 6. Conceptual three-objective design space (area, delay, power) for the optimization loop.

8.1 Baselines

The baselines used for comparison are defined in Table 4.

Table 4. Search baselines and fair-comparison requirements.

Baseline	Definition	Fair-comparison requirement
Exhaustive search	Evaluate all legal configurations where the search space is small enough.	Same constraints and measurement flow; report complete or partial coverage.
Random search	Sample legal configurations randomly.	Same initial information and expensive-evaluation budget; record seeds.
Manual selection	Expert-selected architectures/configurations using documented rules.	Document selection rules before observing test results.
Non-feedback ML	Use a surrogate trained on the initial dataset but do not append new measurements during search.	Same initial dataset, candidate budget, and model protocol.
Active ML + NSGA-II	Use predicted objectives to select batches, then append verified measurements and update the model.	Report all iterations, measured evaluations, failures, and stopping condition.

8.2 Stopping rule

The stopping rule must be fixed before the comparative experiments. A defensible protocol can stop when the maximum number of expensive implementation evaluations is reached, when a predeclared

improvement threshold is not exceeded over a specified number of consecutive batches, or when the available compute budget is exhausted. The final paper must state the exact rule and apply comparable stopping conditions to all baselines. A stopping decision based only on predicted convergence is not sufficient; the reported final Pareto set is based on measured candidates. Templates for the convergence and Pareto-front comparisons are shown in Figure 7 and Figure 8.

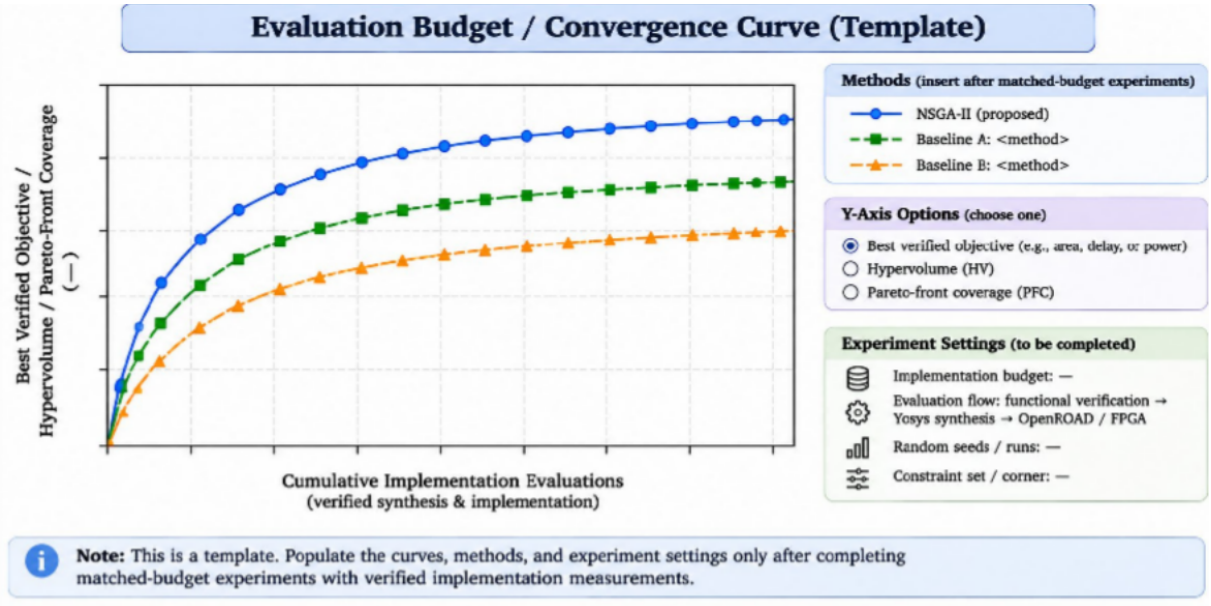


Figure 7. Evaluation-budget/convergence plot template.

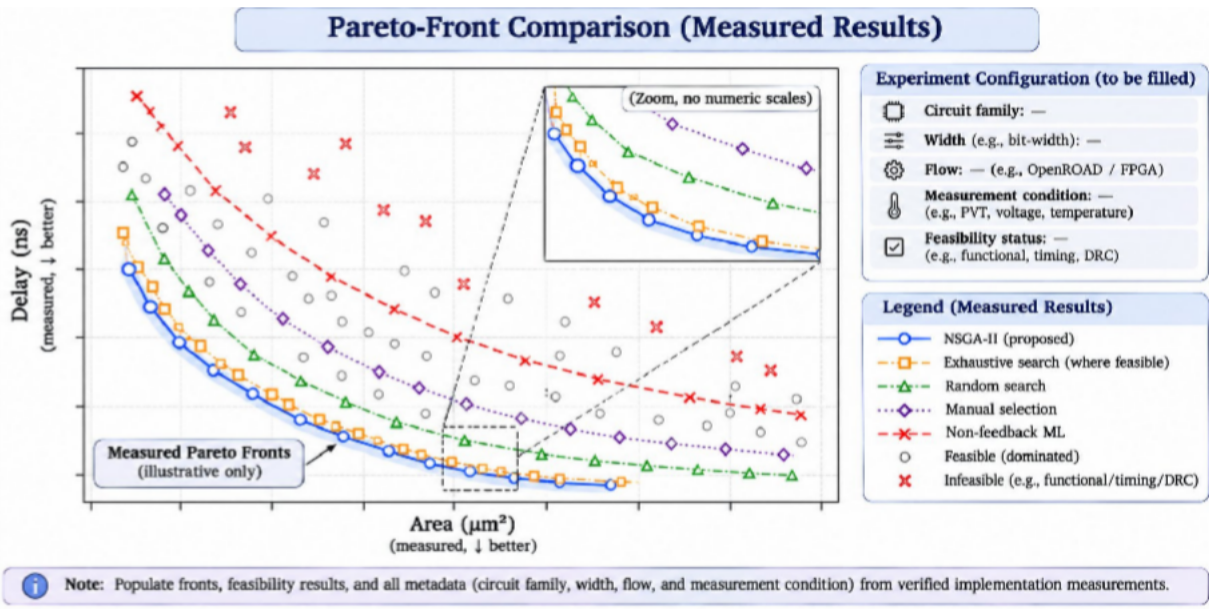


Figure 8. Pareto-front comparison template.

9 Experimental methodology

The experiment campaign is divided into six studies. The values below are protocol fields to be finalized, not claims that experiments have already been performed. Any sample count, random seed, evaluation budget, or split ratio must be selected before execution and then recorded exactly.

The six planned experiments are summarized in Table 5.

Table 5. Planned experiments and protocol status.

Experiment	Purpose	Primary outputs	Protocol status
E1: Functional verification	Compare generated circuits with trusted arithmetic/logical reference models; use directed tests, exhaustive small-width checks, randomized tests, and formal	Pass/fail, tested vectors, coverage description, failed cases, verification runtime.	Protocol to finalize; no results available.
E2: Model comparison	Compare Random Forest, XGBoost, ANN, and a simple baseline on identical leakage-aware splits.	MAE, RMSE, R^2, ranking quality, uncertainty calibration, runtime.	Models proposed; scores pending.
E3: Cross-family / cross-width generalization	Evaluate within-family, leave-one-width-out, and leave-one-family-out transfer.	Held-out errors, feasibility classification, ranking degradation.	Split matrix pending.
E4: Search comparison	Compare active NSGA-II with exhaustive (where feasible), random, manual, and non-feedback ML baselines.	Measured hypervolume, Pareto coverage, feasible yield, evaluations, runtime.	Matched budget and stopping rule pending.
E5: Feature / feedback ablation	Remove feature groups and measured-feedback updates under controlled conditions.	Change in prediction quality and measured search quality.	Ablation matrix pending.
E6: Robustness / reproducibility	Repeat selected runs across seeds and implementation conditions; archive provenance.	Mean, standard deviation, confidence intervals, sensitivity, repeatability.	Replicate count and factors pending.

9.1 Verification protocol

For combinational arithmetic circuits, the testbench shall cover operand boundaries, carry propagation patterns, all-zero/all-one inputs, alternating patterns, signed and unsigned cases where supported, and overflow behavior. Exhaustive tests are feasible for small widths and should be used when the input space permits. For larger widths, combine directed vectors, reproducible randomized vectors, and formal equivalence or assertions where available. Sequential circuits additionally require reset, state-transition, enable, and cycle-level checks.

9.2 Measurement protocol

Every PPA result must identify the metric definition and report source. Area must specify whether it is mapped cell area, placed cell area, core area, FPGA LUT count, or another measure. Delay must specify the path type, timing corner, clock constraint, and whether it is pre- or post-route. Power must state whether it is dynamic, leakage, total, vector-based, vectorless, or tool-estimated, and must record activity assumptions, voltage, temperature, and operating conditions. A timing report with positive slack does not by itself establish a maximum operating frequency unless the clock period and timing methodology support that interpretation.

9.3 Reproducibility and provenance

For every run, archive the RTL source and hash, generator version, configuration file, tool versions, scripts and hashes, technology files or device identity, constraints, random seeds, environment lock file, raw logs, reports, return codes, timestamps, and post-processing scripts. If proprietary technology data cannot be redistributed, record its exact identity and version where permitted. A dataset row without a resolvable provenance manifest must not be used as a verified training label.

10 Results reporting template

No empirical results are available in the source material supplied for this revision. Therefore, this manuscript does not report model accuracy, PPA improvement, evaluation savings, or architectural superiority. Table 6, Table 7 and Table 8 are structured for completion after experiments.

Table 6. Results record template.

Record field	Measured result	Prediction	Verification / provenance
Configuration	Family: [ ]; architecture: [ ]; width: [ ]; ASIC/FPGA flow: [ ].	Model ID: [ ]; predicted area/ delay/power: [ ]; interval: [if calibrated].	Functional status: [ ]; feasibility: [ ]; source hash: [ ]; report path: [ ].
Area	[value + unit + report reference]	[value + unit + model version]	[measured after synthesis/ implementation stage].
Delay	[value + unit + timing corner]	[value + unit + model version]	[path type and timing report].
Power	[value + unit + estimation method]	[value + unit + model version]	[activity assumptions and power report].
Derived metrics	PDP: [ ]; energy/ operation: [only if defined].	Derived prediction only if inputs are predicted.	Do not conflate PDP and energy.

Table 7. Model-comparison template.

Model	Target	MAE	RMSE	R^2	Test split / n	Uncertainty calibration
Random Forest	Area / Delay / Power	TBD	TBD	TBD	TBD	TBD
XGBoost	Area / Delay / Power	TBD	TBD	TBD	TBD	TBD
ANN	Area / Delay / Power	TBD	TBD	TBD	TBD	TBD
Baseline model	Area / Delay / Power	TBD	TBD	TBD	TBD	Not applicable / TBD

Table 8. Search-comparison template.

Search method	Budget	Measured hypervolume	Feasible yield	Pareto coverage	Runtime
Active ML + NSGA-II	TBD	TBD	TBD	TBD	TBD
Random search	Matched TBD	TBD	TBD	TBD	TBD
Manual selection	Matched TBD	TBD	TBD	TBD	TBD
Non-feedback ML	Matched TBD	TBD	TBD	TBD	TBD
Exhaustive search	Where feasible	TBD	TBD	TBD	TBD

11 Discussion framework

The discussion must distinguish predictive performance from optimization performance. A model may achieve low average prediction error without selecting candidates that improve the measured Pareto front. Conversely, a search strategy may identify useful candidates despite moderate average error if its ranking is effective in regions that matter to feasibility and trade-offs. Both claims require measured evidence.

If the active method reduces implementation evaluations, report the exact baseline, matched budget, stopping rule, initial dataset, and measured quality metric. If it does not, report the negative result and investigate whether errors arise from sparse coverage, feature leakage, poor calibration, model extrapolation, constraint handling, or insufficient exploration. No claim that one adder topology is universally superior should be made from a single width or technology condition.

12 Threats to validity and limitations

- **Technology dependence:** learned relationships may not transfer across standard-cell libraries, process corners, FPGA families, or tool versions.
- **Power-estimation uncertainty:** activity assumptions, vector coverage, and tool methodology can materially affect power labels.
- **Dataset bias:** narrow architecture or width coverage can cause poor extrapolation and misleading average metrics.
- **Data leakage:** post-implementation features or related configurations in multiple splits can inflate test performance.
- **Functional coverage:** simulation cannot prove correctness for all possible inputs unless exhaustive or formal methods cover the full state/input space.
- **Search bias:** surrogate-guided selection may repeatedly sample regions the model already understands and miss useful alternatives.
- **Metric ambiguity:** area, delay, power, energy, PDP, and Fmax must have explicit definitions and measurement boundaries.
- **Reproducibility:** unavailable proprietary libraries or incomplete tool manifests may limit independent reproduction.

The framework does not remove the need for synthesis or physical implementation. Its purpose is to prioritize those evaluations. Predicted values remain estimates conditional on the training data and flow; final conclusions must be based on verified implementation outputs.

13 Conclusion

This paper specifies an implementation-aware machine-learning framework for PPA-oriented VLSI design-space exploration. It connects parameterized RTL generation, functional verification, synthesis, physical implementation, feature and provenance management, surrogate modeling, NSGA-II candidate selection, and measured feedback. It defines separate ASIC and FPGA tracks, distinguishes pre-implementation inputs from post-implementation measurements, and prevents model predictions from being treated as verified labels or final Pareto results.

The central empirical question remains open: whether active surrogate-guided search can reduce expensive implementation evaluations while preserving feasible measured Pareto-front quality across circuit families, widths, and implementation conditions. Answering that question requires execution of the proposed experiments, completion of the platform and constraint definitions, and publication of traceable measured results. No superiority or efficiency claim is made until those results exist.

References

[1] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*. 2002;6(2):182–197. https://doi.org/10.1109/4235.996017

[2] Breiman L. Random forests. *Machine Learning*. 2001;45:5–32. https://doi.org/10.1023/A:1010933404324

[3] Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: *Proc*. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016:785–794. https://doi.org/10.1145/2939672.2939785

[4] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. *Neural Networks*. 1989;2(5):359–366. https://doi.org/10.1016/0893-6080(89)90020-8

[5] Ajayi T, Chhabria VA, Fogaça M, Hashemi S, Hosny A, Kahng AB, et al. Toward an open-source digital flow: first learnings from the OpenROAD project. In: *Proceedings of the 56th Annual Design Automation Conference (DAC)*. 2019. Art. 76, p. 1–4. https://doi.org/10.1145/3316781.3326334

[6] YosysHQ. *Yosys Open SYnthesis Suite: documentation* [Internet]. [cited 2026 Nov 12]. Available from: https://yosyshq.readthedocs.io/

[7] OpenROAD Project. *OpenROAD documentation* [Internet]. [cited 2026 Nov 12]. Available from: https://openroad.readthedocs.io/

[8] Pedregosa F, et al. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*. 2011;12:2825–2830.

[9] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: a next-generation hyperparameter optimization framework. In: *Proc*. 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2019:2623–2631. https://doi.org/10.1145/3292500.3330701