MACHINE LEARNING

Machine Learning-Guided Multi-Objective Optimization of Carry-Based Adder Architectures for High-Performance Digital Systems

Priya Singh, Partap Singh

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

Published November 2026

Abstract

This paper presents a verification-first, implementation-aware research protocol for studying machine-learning-guided multi-objective optimization of carry-based adder architectures. The planned benchmark comprises Ripple Carry (RCA), Carry Lookahead (CLA), Carry Select (CSLA), Kogge–Stone, Brent–Kung, and parameterized hybrid-prefix structures under a common arithmetic interface and a declared set of operand widths. The protocol connects parameterized RTL generation, functional verification, implementation-specific synthesis and timing analysis, provenance-preserving dataset construction, surrogate modeling, and budget-matched multi-objective search. Random Forest, gradient-boosted trees, and neural-network regressors are treated as candidate models rather than presumed winners. Functional validity and successful completion of the declared implementation flow are eligibility requirements. FPGA and ASIC experiments are specified as separate tracks, with platform-specific metrics and constraints. The protocol defines leakage-resistant grouped evaluation, uncertainty and calibration checks, measured Pareto-front validation, matched-budget baselines, and reproducibility requirements. No numerical PPA measurements, model scores, architecture rankings, or search improvements are claimed because the implementation campaign has not yet been executed.

Keywords: FPGA; machine learning; NSGA-II; PPA; surrogate modeling; Verilog; VLSI design-space exploration

1 Introduction

Binary addition is a basic operation in processors, digital signal-processing datapaths, cryptographic units, address-generation logic, counters, and arithmetic logic units. In an n-bit adder, the carry dependency couples adjacent bit positions. The architecture used to resolve this dependency affects logic depth, fanout, interconnect, switching activity, area, and ultimately timing and power. Consequently, adder selection is a technology-dependent multi-objective design problem. The architectural and physical trade-offs among prefix networks depend on logic depth, fanout, and wiring, rather than logic depth alone [11], [14].

A Ripple Carry Adder (RCA) propagates carry sequentially and offers a regular structure with low architectural overhead. Carry Lookahead (CLA) computes group generate and propagate terms to reduce serial carry dependence. Carry Select (CSLA) speculatively computes results for alternative carry-in values and selects the appropriate result. Parallel-prefix adders transform carry computation into an associative prefix operation. Kogge–Stone is a dense prefix topology with low logical depth but potentially high wiring and node cost; Brent–Kung uses a sparser tree that trades additional

depth for lower wiring complexity. Hybrid structures can interpolate between these characteristics. These are structural expectations, not universal implementation outcomes [1]–[4], [9]–[12].

The objective is to determine whether implementation-aware machine-learning surrogates can reduce the number of expensive implementation runs needed to identify feasible, non-dominated adder configurations. The study does not presume that a specific architecture is best. Its conclusions must be tied to the selected FPGA device or ASIC technology, synthesis and physical-design flow, timing constraints, RTL coding style, and measured evidence. The common adder datapath assumed throughout is shown in Figure 1.

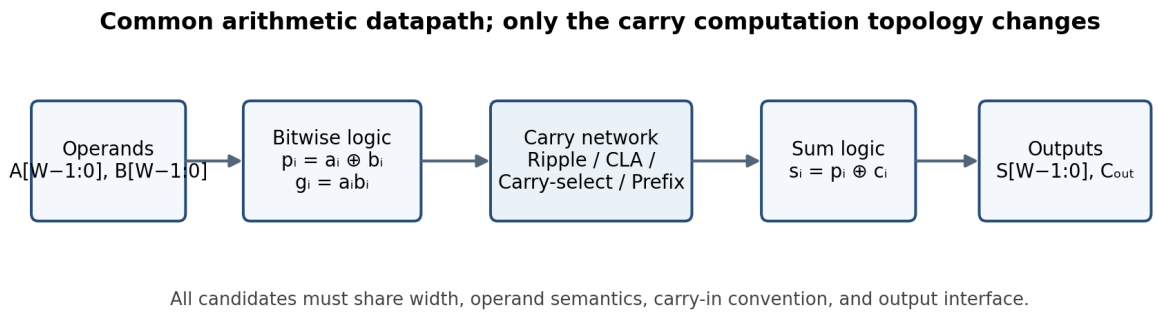


Figure 1. Common adder datapath. The carry network is the architectural variable; the arithmetic interface remains fixed.

2 Background and related work

2.1 Carry generation and propagation

For each bit i, define propagate $p_i = a_i$ XOR b_i and generate $g_i = a_i$ AND b_i. With carry-in c_0, the carry recurrence is $c_{i+1} = g_i$ OR (p_i AND c_i), and the sum bit is $s_i = p_i$ XOR c_i. The same Boolean function can be implemented with serial carry, grouped lookahead, speculative selection, or prefix composition. The exact RTL convention for bit ordering and prefix operand ordering must be stated and tested.

2.2 Prefix computation

The pair (G,P) summarizes a bit group: G indicates that the group generates a carry independent of its input carry, while P indicates that the group propagates an incoming carry through the entire group. Combining a lower group (G_l,P_l) with an upper group (G_u,P_u) gives (G_l OR (P_l AND G_u), P_l AND P_u), under the stated lower-to-upper convention. This operator is associative, enabling parallel-prefix networks. Ladner and Fischer formalized parallel-prefix computation [2]; Brent and Kung studied regular parallel-adder layouts [3]; Kogge and Stone presented recursive-doubling methods for recurrence computation that underpin the dense prefix family [1]. The Sklansky and Han–Carlson families provide additional points in the prefix-network design space [9], [10], while broader comparisons discuss the implementation trade-offs among these structures [11], [12]. The group-combine operation is illustrated in Figure 2.

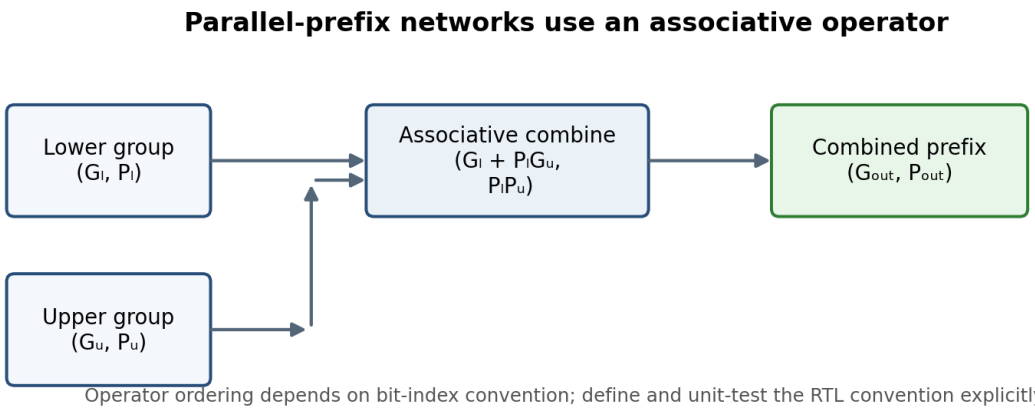


Figure 2. Associative group-combine operation used to construct prefix carry networks.

2.3 Research motivation

The classical topology literature explains logical depth and structural cost, but physical implementation can reorder, optimize, duplicate, or restructure logic. A topology's analytical depth therefore does not directly determine the final delay, area, or power. The 2018 comparison by Balasubramanian and Mastorakis provides a relevant example of architecture-level comparison between CLA and CSLA, while the present protocol extends the comparison to multiple prefix families and a reproducible ML-guided search [4]. Multi-objective evolutionary optimization is motivated by NSGA-II [5].

2.4 Research gap

The targeted gap is a single benchmark methodology that (i) fixes arithmetic semantics across architectures, (ii) records implementation provenance and verification evidence, (iii) prevents data leakage in ML evaluation, (iv) compares surrogate-guided search with budget-matched baselines, and (v) validates predicted Pareto candidates through implementation. The contribution is methodological and experimental; it is not a claim that ML necessarily outperforms conventional search.

3 Objectives, research questions and contributions

Objectives: (1) generate and verify six adder architecture groups at four widths; (2) characterize implementation-dependent PPA and feasibility; (3) compare Random Forest, XGBoost, and ANN surrogate models; (4) compare NSGA-II-guided exploration with exhaustive, random, manual, and fixed-surrogate baselines under matched budgets; and (5) evaluate generalization across held-out widths and architecture families. The research questions are listed in Table 1.

Table 1. Research questions and the evidence required.

Research question	Operational question / evidence
RQ1	How do architecture and width affect measured delay, resource use or area, power, and feasibility?
RQ2	How accurately do RF, XGBoost, and ANN predict each implementation target on leakage-resistant held-out data?

Research question	Operational question / evidence
RQ3	Does ML-guided search improve measured Pareto coverage or hypervolume per implementation evaluation versus baselines?
RQ4	Do models generalize to widths or architecture families excluded from training?
RQ5	How sensitive are results to device, constraints, seeds, tool versions, and power-estimation assumptions?

Contributions: C1, a common arithmetic benchmark interface; C2, a verified architecture set and feature/provenance schema; C3, a comparative surrogate-modeling protocol; C4, a measured-feedback NSGA-II loop; and C5, reporting templates and reproducibility requirements. These contributions become empirical claims only after the experiments are run.

4 Adder architecture definitions and expected structural trade-offs

The six architecture families, their carry strategies and expected trade-offs are summarized in Table 2, and conceptual topology sketches are given in Figure 3.

Table 2. Adder architectures and expected structural trade-offs.

Architecture	Carry strategy	Expected structural characteristic	Potential implementation cost / caveat
Ripple Carry (RCA)	Each stage uses the previous carry.	Simple, regular, low wiring complexity.	Carry path can grow with width; synthesis may map it efficiently to dedicated carry resources on FPGA.
Carry Lookahead (CLA)	Group generate/propagate equations compute carries.	Reduces serial dependence through group logic.	Group size and hierarchy affect fan-in, fanout, and routing.
Carry Select (CSLA)	Compute alternatives for carry-in 0 and 1, then mux.	Reduces carry-dependent selection depth.	Duplicates logic and can increase area and switching.
Kogge–Stone (KS)	Dense parallel-prefix tree.	Low prefix depth and bounded fanout tendency.	Many prefix nodes and long horizontal wiring can raise routing/ resource cost.
Brent–Kung (BK)	Sparse up-sweep/down-sweep prefix tree.	Lower node/wiring demand than dense trees, with greater depth.	Additional levels may increase timing relative to dense prefix.
Hybrid / other prefix	Parameterized combinations of dense/ sparse levels or grouped carry.	Potential intermediate point in the trade-off space.	Must be precisely defined; avoid treating unrelated hybrids as one architecture.

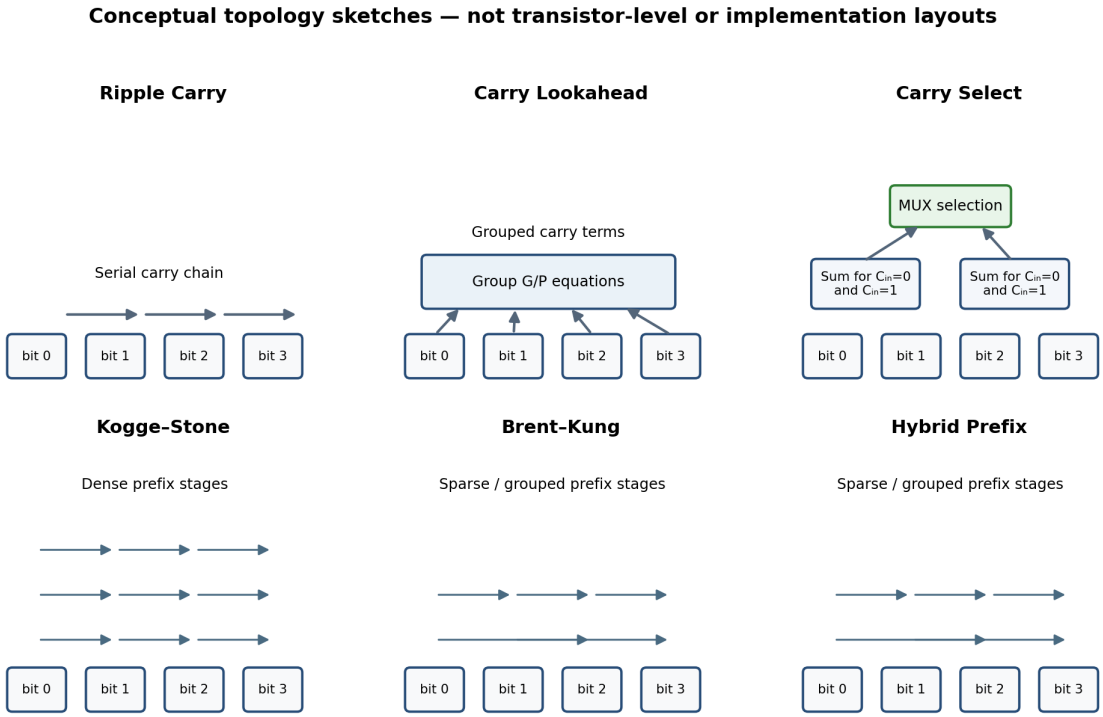


Figure 3. Conceptual topology sketches for six architecture families. These are explanatory diagrams, not synthesized netlists.

4.1 Mathematical relations

The Boolean relations used by all architectures are defined in Table 3.

Table 3. Boolean relations for carry generation and prefix composition.

Quantity	Definition	Meaning
Propagate	$p_i = a_i \oplus b_i$	Bit position propagates an incoming carry.
Generate	$g_i = a_i \cdot b_i$	Bit position generates a carry independent of c_i.
Carry	$c_{i+1} = g_i + p_i c_i$	Equivalent carry recurrence. Here + denotes OR and multiplication denotes AND.
Sum	$s_i = p_i \oplus c_i$	Sum bit.
Group propagate	$P_{j:i} = \prod_{k=i}^{j} p_k$	All bits in the group propagate.
Group generate	$G_{j:i} = g_j + p_j g_{j-1} + \ldots + (\prod_{k=i+1}^{j} p_k) g_i$	Group generates a carry independently of its input.
Prefix combine	$(G_l, P_l) \circ (G_u, P_u) = (G_l + P_l G_u, P_l P_u)$	Associative group composition under the lower-to-upper convention.

Notation note: Boolean OR is written as + and Boolean AND as multiplication. The prefix-combine equation is convention-sensitive. An implementation may store the upper group first or reverse bit indexing; the RTL and reference model must use one convention consistently. A unit test for a two-bit and four-bit group is mandatory.

4.2 Arithmetic specification

The primary benchmark should be unsigned W-bit addition with carry-in: {Cout, Sum} = A + B + Cin, where A and B are W-bit unsigned operands, Cin is one bit, Sum is W bits, and Cout is the carry-out. If signed two's-complement interpretation is also studied, it must be reported as a separate semantic mode: the bitwise adder is shared, but signed overflow is not equivalent to Cout. Define overflow as V = carry_into_MSB XOR carry_out_of_MSB for two's-complement addition. Do not mix signed and unsigned result labels without recording the mode.

All designs must use the same clocking, reset policy (preferably none for a purely combinational adder), port widths, latency, and timing constraints. If a registered/pipelined version is introduced, it forms a separate benchmark because latency and throughput differ.

5 Problem formulation and objective hygiene

Represent a candidate by x = [W,T,H,F], where W is width, T is topology, H contains topology parameters (group size, radix, sparsity, block partition, or hybrid composition), and F contains implementation-flow settings. The primary minimization vector is f(x) = [D(x), A(x), P(x)] when delay, area/resource use, and power are all available on a consistent basis. For FPGA, A may be a vector or a declared scalarization of LUTs, FFs, carry resources, BRAM, and DSP usage; preserve the individual resource counts even if a scalar score is used.

Power-delay product is PDP(x) = P(x) × D(x). PDP is a derived metric and is not energy. Energy requires an explicitly defined observation interval and boundary, e.g., $E = \int_0^T P(t)dt$ or average power multiplied by a justified execution interval. For a combinational adder, energy per operation requires a workload/activity model and operation rate; it cannot be inferred from PDP alone. Do not optimize both P, D, and PDP as independent objectives without explaining redundancy and the intended decision context. The distinction between architecture, circuit family, and wiring effects is also emphasized in logical-effort analysis of carry-propagate adders [14].

Functional validity V(x)=1 is a hard constraint. Other constraints may include maximum delay, resource ceilings, power ceiling, timing closure, and successful completion of the implementation flow. Define limits before evaluating candidates. Feasible Pareto membership is computed only from measured, functionally valid candidates using declared objectives and constraints. Predicted values can guide search but cannot establish final Pareto membership.

A supplementary scalar score may be used for ranking:

$$J(x) = w_D D_{\text{norm}} + w_A A_{\text{norm}} + w_P P_{\text{norm}} + \lambda \cdot \text{Penalty}(x) \quad (1)$$

Weights, penalty magnitude, and normalization bounds must be specified before the search, justified by the application, and frozen before held-out evaluation. Report sensitivity to alternative reasonable weights; do not select weights after inspecting held-out outcomes.

6 Research workflow and candidate generation

The workflow comprises parameterized RTL generation, equivalence-oriented functional verification, synthesis and implementation, provenance-aware dataset construction, model training, multi-objective search, and measured feedback. FPGA and ASIC-oriented experiments are separate tracks. FPGA results may use AMD Vivado reports for a specified device; ASIC results may use Yosys and

OpenROAD with a fully specified technology platform, libraries, constraints, and physical-design configuration. The two tracks must not be pooled as if LUTs and standard-cell area were interchangeable. The closed-loop workflow is shown in Figure 4.

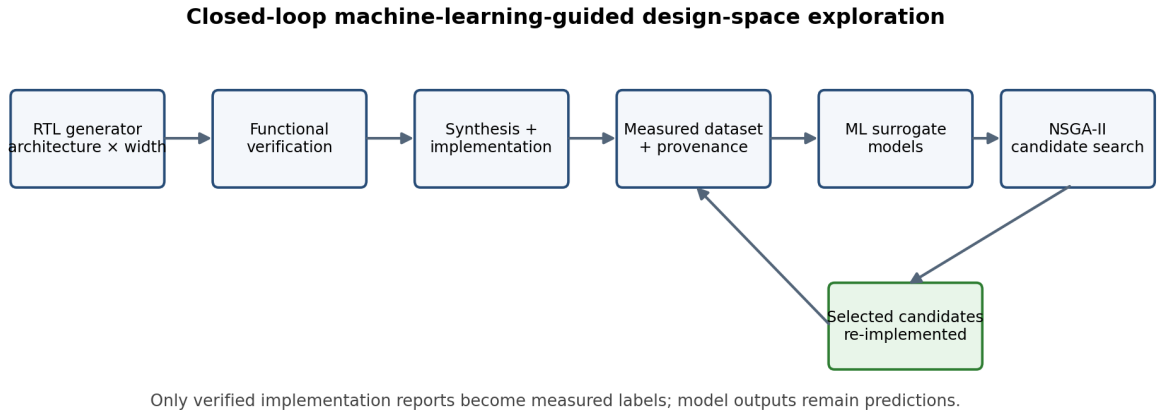


Figure 4. Machine-learning-guided closed-loop workflow. The feedback path represents re-implementation and dataset update; only implementation reports are measured labels.

6.1 Candidate generation

- Enumerate $W \in \{8,16,32,64\}$; architecture family; and declared topology parameters. Document whether every family is implemented at every width.
- Use a deterministic generator and retain source hash, generator version, parameter manifest, and elaborated RTL.
- Ensure each candidate is synthesizable and avoid architecture-specific behavior that changes arithmetic semantics.

7 RTL design and functional verification

7.1 Module interface

A reference interface is: inputs [W−1:0] A and B, input Cin, outputs [W−1:0] Sum and Cout. Parameter W must be elaboration-safe for all target widths. The combinational module should use complete assignments and avoid inferred latches. The architecture-specific implementation may expose internal p/g signals for debug, but public I/O semantics must remain identical.

7.2 Test matrix

The functional test matrix applied to every candidate is given in Table 4.

Table 4. Functional verification test matrix.

Test class	Example stimulus	Purpose
Zero / identity	A=0, B=0; A arbitrary, B=0	Basic arithmetic and identity.
Maximum values	A=2^W−1, B=1, Cin=0	Full-width carry propagation and Cout.
Carry-in	Cin=1 with zero and nonzero operands	Correct initial carry handling.

Test class	Example stimulus	Purpose
Alternating patterns	A=0101..., B=1010...	Bit-level propagate/generate patterns.
Single-bit cases	A=2^i, B=2^j	Isolated generate and carry paths.
Long carry chain	A=all ones except selected bit; B chosen to propagate	Stress carry propagation.
Randomized	Seeded pseudo-random A, B, Cin	Broader input coverage.
Boundary widths	W=8,16,32,64 elaboration and edge bit positions	Parameter and indexing correctness.

7.3 Verification acceptance

A candidate is eligible for PPA analysis only if it elaborates, passes the declared functional suite, and produces a traceable verification record. Failed candidates remain in the dataset with status=verification_failed; excluding them silently would bias the reported feasible-candidate rate. For small W, exhaustive enumeration over 2^(2W+1) input combinations can be performed; for W=8 this is 131,072 combinations, which is practical for simulation. For larger widths, use formal equivalence if available and document proof assumptions.

8 FPGA/ASIC implementation and PPA measurement

8.1 Platform configuration, implementation and power measurement

For FPGA, record the exact part number, speed grade, package if relevant, Vivado version, synthesis strategy, implementation strategy, constraints, clock period, I/O delays, operating conditions, and whether power is vectorless or activity-based. Preserve LUT, FF, carry-chain, BRAM, DSP, timing, Fmax, and power report values. Report whether timing is post-synthesis, post-placement, or post-route.

For ASIC, record the process/PDK, standard-cell library and corner, operating voltage and temperature, Yosys version and scripts, OpenROAD version, floorplan/utilization, clock constraints, placement/routing settings, parasitic extraction method, and timing/power report stage. State whether area is cell area, core area, or another measure. Power must identify whether it is dynamic, leakage, or total and specify activity annotation.

Power estimation is sensitive to switching activity, clocking, glitches, capacitance, and tool methodology. If only estimated power is available, label it as estimated. Compare candidates only when the power methodology and activity assumptions are consistent.

9 Dataset construction and leakage control

Every record must link a configuration to exact RTL and tool provenance. Separate pre-implementation features from post-implementation labels. A predictor intended to estimate final delay must not receive final delay, routed wire length, post-route slack, or other downstream quantities as input. The field groups of each dataset record are defined in Table 5.

Table 5. Dataset field groups and their roles.

Field group	Candidate fields	Role / caution
Identity	candidate_id, source_hash, generator_version, timestamp, replicate_id	Unique provenance; do not use identifiers as predictive shortcuts.
Architecture	width, topology, group_size, radix, prefix_levels, sparsity	Pre-implementation features if derived from RTL/ configuration.
Boolean structure	count of p/g nodes, estimated logic depth, prefix-node count, fanout estimate	Define extraction algorithm and validate against elaborated netlist.
Flow settings	device/library, tool version, clock target, constraints, synthesis options	Context features; generalization depends on how these are split.
Verification	simulator, seed, vectors, test classes, pass/fail	Eligibility and provenance; do not let test outcome leak into continuous PPA prediction unless explicitly modeling feasibility.
Measured labels	LUT, FF, carry resources, cell area, delay, slack, Fmax, power	Targets from named reports; stage and units required.
Derived labels	PDP=P×D; energy only with interval/ activity definition	Derive after label validation; do not confuse with independent measurements.
Outcome status	verified, failed, infeasible, tool_failed, incomplete, estimated	Retain all outcomes and state missingness reasons.

9.1 Split strategy

At minimum, use grouped train/validation/test splits. A stronger transfer experiment is leave-one-width-out or leave-one-architecture-family-out: train on a subset of widths/families and evaluate on the held-out group. With only four widths and six families, report the small number of groups and avoid overgeneralizing. Repeated implementation runs of the same candidate must remain in one partition. If multiple configurations are near-duplicates generated from the same RTL template, group them where appropriate. The partitioning scheme is shown in Figure 5.

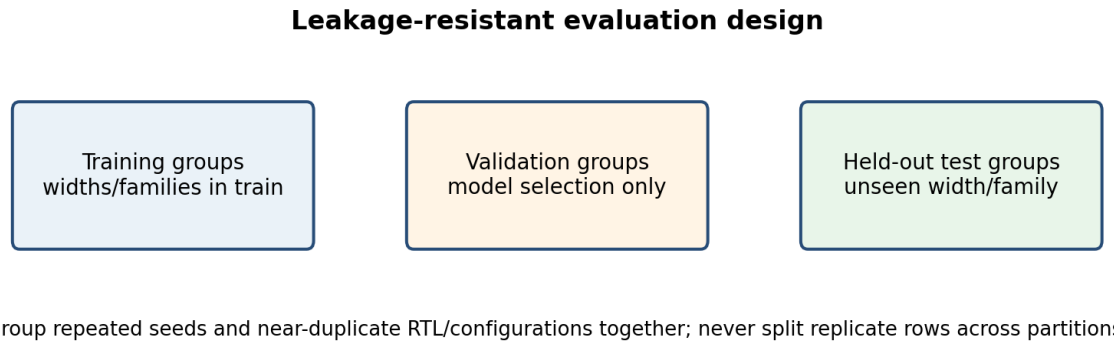


Figure 5. Leakage-resistant data partitioning. Related candidates are grouped before splitting; the held-out partition is used once for final evaluation.

9.2 Dataset quality

Report number of generated candidates, functionally valid candidates, completed implementations, failed runs, infeasible runs, repeated measurements, missing labels by target, and unique architecture/width combinations. Provide descriptive statistics and distributions before model training. Do not impute failed implementation outcomes as ordinary numeric PPA labels.

10 Predictive modeling and evaluation

For target k, the surrogate is $\hat{y}_k(x)=M_k(\varphi(x);\theta_k)$, where $\varphi(x)$ contains permitted pre-measurement features and θ_k contains learned parameters. Random Forest is a tree-ensemble baseline [6]; XGBoost is a gradient-boosted tree method [7]; an ANN is a flexible nonlinear approximator whose capacity must be controlled relative to the dataset size [8]. The configuration details to be reported for each model are listed in Table 6.

Table 6. Candidate surrogate models: configuration details and controls.

Model	Required configuration details	Risks / controls
Random Forest	Tree count, max depth, max features, minimum leaf size, bootstrap, seed	Can overfit small grouped datasets; report feature importance cautiously.
XGBoost	Estimators, depth, learning rate, subsampling, regularization, early stopping, seed	Tune on validation only; avoid excessive capacity.
ANN	Layer widths, activation, optimizer, learning rate, batch size, epochs, regularization, seed	Requires scaling and careful early stopping; report parameter count.
Optional baseline	Dummy mean/median predictor and simple linear/ridge model	Shows whether nonlinear models add value.

Metrics are calculated per target and per split:

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

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

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

Report units for MAE/RMSE. R^2 may be negative on held-out data and should not be clipped. For strongly skewed targets, a log transform may be evaluated but the transformation and inverse-transform procedure must be stated.

Use repeated grouped cross-validation when dataset size permits. Report mean and standard deviation across folds or seeds, and preferably confidence intervals with a justified resampling method. Do not treat individual correlated candidate rows as independent replicates. For model selection, do not tune against the final held-out test set.

11 Multi-objective optimization and measured feedback

NSGA-II ranks candidate solutions using non-dominated sorting and crowding distance, maintaining a diverse approximation to the Pareto front [5]. For this study, decision variables include architecture family, width (when width is a study factor rather than fixed), group size, radix, sparsity, and hybrid composition. Discrete and categorical operators are required for architecture variables; invalid combinations must be rejected or repaired deterministically.

The primary objectives should be limited to metrics with comparable and valid measurement semantics. A practical primary configuration is minimize [delay, area/resource score, power]. PDP may be reported as a secondary derived metric. If energy is included, define the workload, activity, voltage/frequency, measurement interval, and energy boundary. The NSGA-II settings to be reported are listed in Table 7.

Table 7. NSGA-II parameters and reporting requirements.

NSGA-II parameter	What must be reported
Population size	40 individuals; assumed mixed discrete/continuous search population.
Generations / stopping	50 generations; stop at 100 implementation evaluations or after 10 generations with <1% hypervolume improvement.
Crossover / mutation	Operator type, probabilities, distribution indices or discrete mutation rules.
Constraint handling	Feasibility-first or documented penalty; hard functional validity gate.
Random seeds	All seeds and number of independent runs.
Surrogate usage	Which models predict which objectives; model version/hash.
Candidate validation	How many predicted Pareto candidates are re-implemented and how selected.
Baseline budget	Same number of implementation evaluations for random/manual/active search comparisons.

The measured Pareto front is computed from all measured feasible candidates under the declared objectives. The predicted Pareto front is plotted separately. Compare them using hypervolume with a fixed reference point, coverage, spacing/diversity, and evaluation count. Reference-point selection must be declared before final test evaluation. A conceptual three-objective Pareto view is shown in Figure 6.

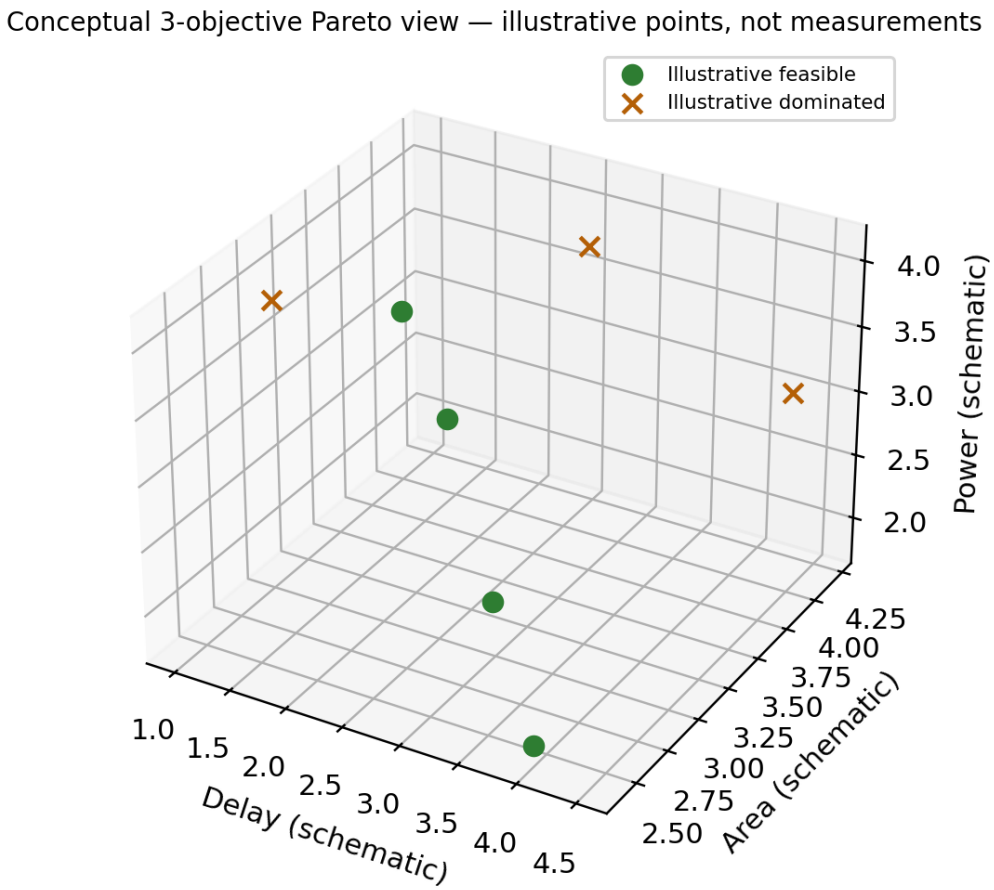


Figure 6. Conceptual three-objective Pareto illustration. Points are schematic placeholders and are not experimental observations.

11.1 Uncertainty-aware candidate selection

A recommended extension is uncertainty-aware selection: query candidates with high predicted Pareto value, high model disagreement, or high uncertainty. Uncertainty estimates must be calibrated on validation data; do not assume a fixed percentage error band.

12 Experimental design and evaluation plan

The planned experiments are summarized in Table 8 and the search baselines in Table 9.

Table 8. Planned experiments.

Experiment	Design	Primary outputs
E1: Functional equivalence	All architecture/width combinations; directed, random, and exhaustive small-width tests.	Pass rate, failure categories, verification runtime.
E2: Architecture characterization	Implement verified candidates across widths and topologies.	Measured resources/area, delay, power, feasibility.
E3: Surrogate comparison	Grouped splits; RF, XGBoost, ANN plus simple baseline.	MAE, RMSE, R^2, residual analysis by target.
E4: Transfer	Leave-one-width-out and leave-one-family-out.	Generalization gap and failure modes.

Experiment	Design	Primary outputs
E5: Search comparison	Matched implementation budgets for NSGA-II, random, manual, fixed surrogate, active feedback.	Measured hypervolume, Pareto coverage, feasible rate, run count.
E6: Ablation	Remove structural features, flow features, active feedback, or uncertainty selection.	Change in prediction and search quality.
E7: Robustness	Repeat seeds; vary permitted flow settings or constraints.	Variance, rank stability, sensitivity.
E8: Cross-flow validation	Separate FPGA and ASIC tracks; compare only within each context unless normalized.	Flow-specific conclusions and transfer limitations.

12.1 Baselines

Table 9. Search baselines and protocols.

Baseline	Protocol
Exhaustive search	Evaluate all candidates in a finite tractable subset; use as reference for that subset only.
Random search	Sample with fixed seeds and same implementation budget.
Manual selection	Pre-register a rule based on topology expectations; do not tune after seeing results.
NSGA-II, no feedback	Use a fixed initial surrogate and do not append newly measured candidates.
Active-feedback ML	Iteratively implement selected candidates and retrain using measured feedback.

12.2 Replication and statistics

For each configuration, determine whether repeated implementation runs are needed to characterize tool nondeterminism. Choose replicate counts after a pilot and report the rationale; do not select a convenient number after observing favorable outcomes. Where multiple seeds are used, report per-seed results and aggregate statistics. Compare search strategies with paired seeds or matched budgets when feasible. Use effect sizes and uncertainty intervals rather than relying only on a single best run.

12.3 Result tables

Results are to be recorded in the template shown in Table 10, with one row per candidate.

Table 10. Results record template (to be completed after the experiments).

Architecture	W	LUT	FF	Carry	Area/unit	Delay/unit	Fmax	Power/type	PDP	Status
TBD	8/16/32/64	TBD	TBD	TBD	TBD	TBD	TBD	TBD	Derived	TBD
One row per candidate	Actual width	Measured	Measured	Measured	Measured/N.A.	Measured	Derived from timing	Measured/estimated	P×D only	Verified/infeasible/fail

13 Results presentation and figure plan

No empirical results are reported in this protocol manuscript. Figures in the methods sections are conceptual illustrations only; all result plots must be generated from verified implementation records after the experimental campaign.

13.1 Required plot hygiene

- Use consistent units and axis limits for direct comparisons; state whether lower or higher is better.
- For FPGA, plot LUT, FF, carry resources, timing/Fmax, and power separately or as clearly labelled panels.
- For ASIC, state cell area versus core area, timing stage, power type, library corner, and voltage/temperature.
- Show measured points distinctly from surrogate predictions. Predicted points must not be labelled as measured.
- Show failed and infeasible candidates in a separate status summary; do not hide them from feasibility accounting.
- Include raw-data availability and plotting-script version in the reproducibility package.

14 Results interpretation and discussion framework

No numerical result is asserted in this manuscript. The expected qualitative hypothesis is that topology effects will vary with width and implementation context. RCA is expected to be a compact baseline but may suffer a long carry path; CLA and prefix networks may reduce logical carry depth but introduce more logic and routing; CSLA may trade duplicated computation for reduced carry-dependent delay; dense and sparse prefix structures may exhibit different area/timing/wiring balances. These statements must be tested rather than written as findings.

The final discussion should answer each RQ using measured evidence. RQ1 should report per-width distributions and feasible candidates. RQ2 should compare model errors by target and held-out group. RQ3 should compare measured Pareto quality per implementation evaluation under equal budgets. RQ4 should report transfer performance and where it fails. RQ5 should quantify sensitivity to seeds, constraints, and tool settings. Avoid claims such as 'Kogge–Stone is fastest' unless supported by the specific measured flow and accompanied by uncertainty and resource trade-offs.

14.1 Evidence categories

The statements permitted for each class of evidence are defined in Table 11.

Table 11. Evidence classes and permitted statements.

Evidence class	Permitted statement	Not permitted
Measured	Values traceable to implementation reports and exact candidate IDs.	Filling TBD cells with estimates or literature values as if measured.
Predicted	Surrogate outputs with model version, split, and uncertainty/residual information.	Calling predictions experimental measurements.
Analytical / conceptual	Boolean equations, topology definitions, asymptotic discussion.	Treating asymptotic depth as guaranteed physical timing.

Evidence class	Permitted statement	Not permitted
Qualitative hypothesis	Expected trade-offs to be tested.	Writing hypothesis as observed result.

15 Threats to validity, limitations and reproducibility

15.1 Construct validity

PPA metrics may not represent application-level performance if workload, activity, clocking, or timing context is unrealistic. FPGA power estimates can differ materially depending on switching activity and implementation stage. Area may mean LUT count, cell area, or core area; these are not interchangeable. Define every metric precisely.

15.2 Internal validity

RTL coding style, synthesis heuristics, constraints, physical-design settings, and tool versions may affect outcomes. Hold the flow constant within each track and record deviations. Use identical constraints and equivalent interfaces. Do not compare candidates synthesized with different optimization settings without treating the setting as a factor.

15.3 External validity

Four widths and six architecture groups do not represent every adder topology, process node, FPGA family, or workload. Hybrid parameter coverage may be incomplete. Generalization claims must be bounded to the tested widths, architectures, devices, and flows.

15.4 Machine-learning validity

Small sample size, family imbalance, correlated variants, feature leakage, distribution shift, and stochastic training can produce optimistic estimates. Use grouped splits, held-out transfer tests, baseline models, ablations, and repeated seeds. Feature importance is correlational, not causal.

15.5 Reproducibility package

The reproducibility package should contain the following:

- RTL source and deterministic generator scripts; candidate IDs and hashes.
- Testbenches, reference model, simulator version, seeds, logs, and optional formal scripts.
- Tool versions, device/technology identifiers, constraints, libraries, corners, and flow scripts.
- Raw synthesis, placement, routing, timing, and power reports; exact report extraction scripts.
- Processed dataset with units, missingness/status fields, and provenance links.
- Model training code, preprocessing, hyperparameters, seeds, model artifacts, and split manifests.
- NSGA-II code, operators, budgets, reference point, seeds, and selected-candidate manifests.
- Plotting code, figure source data, and manuscript revision identifier.

16 Conclusion

This manuscript defines a controlled research protocol for machine-learning-guided multi-objective optimization of carry-based adder architectures. It specifies a common arithmetic interface for RCA,

CLA, CSLA, Kogge–Stone, Brent–Kung, and hybrid structures; a verification-first RTL-to-implementation workflow; provenance-aware dataset construction; leakage-resistant surrogate evaluation; and NSGA-II search with measured feedback. The protocol also defines matched-budget baselines, platform-specific reporting, uncertainty analysis, and a reproducibility package. The study's empirical questions remain unresolved: no architecture is declared superior, no predictive accuracy is claimed, and no reduction in implementation effort is asserted until actual experiments have been completed and independently checked.

References

[1] Kogge PM, Stone HS. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*. 1973;C-22(8):786–793. https://doi.org/10.1109/TC.1973.5009159

[2] Ladner RE, Fischer MJ. Parallel prefix computation. *Journal of the ACM*. 1980;27(4):831–838. https://doi.org/10.1145/322217.322232

[3] Brent RP, Kung HT. A regular layout for parallel adders. *IEEE Transactions on Computers*. 1982;C-31(3):260–264. https://doi.org/10.1109/TC.1982.1675982

[4] Balasubramanian P, Mastorakis N. Performance comparison of carry-lookahead and carry-select adders based on accurate and approximate additions. *Electronics*. 2018;7(12):369. https://doi.org/10.3390/electronics7120369

[5] 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

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

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

[8] 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

[9] Sklansky J. *Conditional-sum addition logic*. IRE Transactions on Electronic Computers. 1960;EC-9(2):226–231.

[10] Han T, Carlson DA. Fast area-efficient VLSI adders. In: *Proceedings of the 8th IEEE Symposium on Computer Arithmetic*. 1987:49–56.

[11] Harris D. A taxonomy of parallel prefix networks. In: *Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers*. 2003;2:2213–2217. https://doi.org/10.1109/ACSSC.2003.1292373

[12] Yagain D, Krishna V, Baliga A. Design of high-speed adders for efficient digital design blocks. *International Scholarly Research Notices*. 2012;2012:253742. https://doi.org/10.5402/2012/253742

[13] Balasubramanian P, Maskell DL. A new carry look-ahead adder architecture optimized for speed and energy. *Electronics*. 2024;13(18):3668. https://doi.org/10.3390/electronics13183668

[14] Harris D, Sutherland I. Logical effort of carry propagate adders. In: *Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers*. 2003.