Graph Neural Network-Guided Design Space Exploration for Power, Performance, and Area Optimization in Advanced VLSI Circuits

Mathijs Genefaas
Department of Computer Science, University of Alabama at Birmingham, Birmingham, AL, USA.
mathijs1989@uab.edu

Paulette Lemaitre
School of Computing, Clemson University, Clemson, SC, USA.
lemaitre57@clemson.edu

Jose Mari Roda
Department of Computer Science and Engineering, University of Nevada, Reno, Reno, NV, USA.
josemari613@unr.edu

Kaja Lusiak
Department of Computer Science, Colorado State University, Fort Collins, CO, USA.
hellokaja@colostate.edu

Abstract

Advanced VLSI circuits at sub-10-nm nodes require design space exploration across discrete and continuous decisions whose effects on power, performance, and area are strongly coupled by circuit topology. Conventional surrogate models treat a candidate configuration as a flat feature vector and therefore lose connectivity, hierarchy, and reconvergent-path information that govern timing and power. This manuscript develops a graph neural network-guided design space exploration framework that represents each candidate circuit as a heterogeneous graph, learns message-passing embeddings of cells, nets, and design variables, and is designed to predict power, performance, and area with calibrated uncertainty. The framework combines a multi-task graph neural network surrogate with constrained Bayesian optimization. A normalized scalarization defines exploration directions, while expected improvement is weighted by constraint-feasibility probabilities for timing, power density, and area budgets. The manuscript formalizes graph construction, message-passing and readout operators, a heteroscedastic predictive model, the training objective, and a constrained acquisition rule. It also analyzes how the framework differs from evolutionary search, Gaussian-process optimization, and tree-ensemble surrogates, and it identifies failure modes related to distribution shift, surrogate exploitation, graph scaling, and constraint correlation. The contribution is a technically explicit methodological foundation for topology-aware power-performance-area exploration that keeps trade-offs visible rather than collapsing them prematurely. The framework is intended to guide reproducible implementation and rigorous evaluation in advanced-node design flows.

Keywords

Graph neural networks; design space exploration; VLSI physical design; power-performance-area; constrained optimization; surrogate modeling

1. Introduction

Advanced VLSI design under scaled process nodes is dominated by coupled power, performance, and area trade-offs. Gate sizing, threshold-voltage assignment, buffer insertion, placement legalization, routing topology, and clock-tree parameters interact through timing paths, power grids, and local congestion. Machine learning has been applied to physical design tasks [1], graph-based learning has been used for placement [2], and broader surveys document the growing role of learning in electronic design automation [3]. These developments motivate models that can reason directly over circuit connectivity rather than over hand-engineered summaries of a design.

Graph neural networks provide a natural model for circuit topology because they operate on relational data [4] and generalize graph convolution [5], with expressiveness characterized by message-passing power [6], gated propagation [7], and neural message passing [8]. A circuit is not a set of independent cells; timing slack propagates along paths, switching activity couples through shared nets, and power density depends on neighborhood structure. A graph representation preserves these relations and supports permutation-equivariant learning. The same representation can incorporate heterogeneous node types such as cells, nets, terminals, and macros, together with design-variable attributes that describe the candidate configuration.

Design space exploration is a multi-objective optimization problem. Evolutionary algorithms such as NSGA-II are widely used [9], Bayesian optimization offers sample-efficient search [10], practical acquisition strategies make such optimization tractable [11], and surrogate modeling provides the underlying approximation logic [12]. Common surrogates include tree ensembles [13], random forests [14], and Gaussian processes [15]. These methods are effective for low-dimensional or fixed-feature problems, but they do not directly encode circuit connectivity, hierarchy, or heterogeneous design variables. A flat feature vector can represent total area or estimated power, yet it discards the path structure that determines whether a local change improves frequency or merely shifts delay to another path. This loss is especially consequential in advanced nodes, where variability, RC parasitics, and multi-patterning constraints make local and global effects difficult to separate.

The central contribution of this manuscript is a graph neural network-guided design space exploration framework for power, performance, and area optimization in advanced VLSI circuits. The framework formalizes a mixed discrete-continuous design space, represents each candidate as a heterogeneous circuit graph, learns a multi-task surrogate with uncertainty estimates, and uses constrained Bayesian optimization to propose new configurations. The methodological development emphasizes why graph message passing is appropriate, how the surrogate is trained without collapsing the three objectives prematurely, how constraints are incorporated into acquisition, and which failure modes must be controlled in practice. The analysis is methodological and analytical rather than an empirical benchmark report; numerical outcomes depend on the process node, library, and implementation flow and are therefore not asserted here.

2. Problem Formulation and Design Representation

A design configuration is a mixed discrete-continuous vector drawn from a structured design space. Continuous variables include transistor widths, gate sizes, wire widths, and buffer strengths. Discrete variables include threshold-voltage classes, cell variants, layer assignments, and legal placement sites. The power-performance-area target is a vector-valued function of the configuration:

$$q(d) = \big(\mathrm{P}(d), F(d), A(d)\big)^{T}$$

where P is power, F is performance expressed as operating frequency, and A is area. Lower P and A are preferred, while higher F is preferred. The superscript T denotes transposition. This vector is not scalarized during prediction; scalarization is deferred to the exploration policy so that trade-offs remain explicit.

To expose circuit structure to the surrogate, each candidate is represented as a heterogeneous graph. For configuration d, the graph is

$$G_d = (V, E, X_V, X_E, d)$$

where V contains cell, net, terminal, and macro nodes; E contains connectivity and hierarchy edges; X_V and X_E are node and edge feature matrices; and d supplies global design-variable attributes. Node features include cell type, drive strength, threshold-voltage class, input capacitance, leakage class, switching activity, and placement coordinates where available. Edge features include net capacitance, resistance estimates, fanout count, and timing-arc attributes. The graph is permutation equivariant, so the representation does not depend on an arbitrary node ordering.

The graph formulation has three advantages for design space exploration. First, it preserves path structure: a timing-critical path is a sequence of connected cells and nets, and message passing can propagate delay and slew information along that sequence. Second, it supports heterogeneous variables: a threshold-voltage assignment can be attached to a cell node, while a global supply-voltage choice can be attached as a graph-level attribute. Third, it scales with local neighborhood operations rather than with a dense all-to-all feature matrix. The graph is not a substitute for signoff analysis; it is an inductive bias for learning a surrogate that respects the relational nature of the circuit.

3. Graph Neural Network Surrogate for Power, Performance, and Area

Node states are updated by a message-passing rule that aggregates local fan-in and fan-out information:

$$h_v\hat{}(l+1) \;=\; \sigma(W_s\hat{}(l)\, h_v\hat{}(l) \;+\; \sum_\{u \in N(v)\}\, W_e\hat{}(l)\, h_u\hat{}(l) \;+\; b\hat{}(l))$$

Here h_v^(l) is the state of node v at layer l, N(v) is its neighborhood, W_s^(l) and W_e^(l) are learned weight matrices for self and neighbor messages, b^(l) is a bias vector, and σ is a nonlinear activation. The sum aggregation preserves equivariance and captures local timing and power coupling across connected cells and nets. Edge features are concatenated to neighbor states before the neighbor transformation, so net capacitance, fanout, and timing-arc attributes influence the message. Stacking L layers allows information to propagate across L-hop neighborhoods, which is sufficient to capture reconvergent paths and local power-density effects in many advanced-node exploration settings.

After L layers, a graph-level embedding is formed by combining mean and max pooling:

$$z_d = (1/|V|) \sum_\{v \in V\} h_v\^(L) \oplus max_\{v \in V\} h_v\^(L)$$

where $\oplus$ denotes concatenation, |V| is the number of nodes, and z_d is the circuit-level representation used for power-performance-area prediction. Mean pooling captures global activity, while max pooling preserves critical local effects such as timing-critical paths or power-dense regions. The embedding is passed through task-specific heads for power, frequency, and area, with shared message-passing parameters to exploit correlation among the three targets.

Surrogate inference may also occur under deployment constraints. When surrogate queries must be outsourced or performed under cryptographic constraints, reduced-activation architectures for two-party private inference are relevant [16]; the present framework does not require such a setting, but the constraint illustrates that surrogate complexity is a deployment variable rather than only a modeling choice. Training uses Adam [17], with batch normalization [18] and dropout [19] used for stabilization and regularization. These choices are standard for graph learning but must be paired with a probabilistic output layer to support uncertainty-aware acquisition.

The surrogate maps z_d to a Gaussian predictive distribution over normalized power-performance-area targets. This follows standard probabilistic modeling practice [20]. The predictive form is

$$p(q \mid d) = N(q;\ \mu_\theta(G_d), diag(\sigma_\theta^2(G_d)))$$

where μ_θ and σ_θ^2 are the mean and variance outputs of the graph neural network with parameters θ, N denotes a multivariate normal distribution, and diag forms a diagonal covariance matrix. The diagonal assumption permits independent uncertainty estimates for power, frequency, and area; cross-objective correlations are still represented through shared message-passing parameters and auxiliary covariance heads if needed.

The training objective is the Gaussian negative log-likelihood over the observed configurations:

$$L(\theta) = \sum_i (q_i - \mu_\theta(G_i))^2/(2\sigma_\theta^2(G_i)) + ½ log\ \sigma_\theta^2(G_i)$$

where q_i is the observed normalized power-performance-area vector for sample i, G_i is the corresponding graph, and the summation runs over the training set. The first term penalizes mean prediction error relative to the predicted variance, while the second term prevents the model from inflating variance to reduce the first term. In practice, a small variance floor is used for numerical stability. The output heads are trained jointly because power, frequency, and area are physically coupled: a sizing change that improves frequency may increase both area and leakage power, and the surrogate must learn these trade-offs rather than treat the targets as independent regressions.

The graph neural network surrogate is appropriate relative to credible alternatives because it encodes topology explicitly. A random forest or gradient-boosted tree model can fit a tabular representation of aggregate features, but it cannot distinguish two circuits with identical aggregate capacitance and area yet different path distributions. A Gaussian process with a standard kernel can provide calibrated uncertainty in low-dimensional spaces, but its scaling and kernel design become problematic when the input includes graph-structured variables. A graph neural network does not eliminate the need for feature engineering, but it shifts the engineering effort from hand-crafted path summaries to graph construction and message-passing design.

4. Constrained Multi-Objective Design Space Exploration

For a given weight vector λ = (w_P, w_F, w_A) with nonnegative weights summing to one, a normalized scalarization is

$$J_\lambda(d) = w_P\, P(d)/P_0 + w_A\, \hat{A}(d)/A_0 + w_F\, (1 - F(d)/F_0)$$

where $\hat{P}$, $\hat{F}$, and $\hat{A}$ are surrogate predictions, and P_0, F_0, A_0 are reference scales derived from the current design or specification. Minimizing J_λ favors low power and area and high frequency. The scalarization is not used to collapse the problem permanently; it defines a query direction for constrained Bayesian optimization. By varying λ, the designer can explore the Pareto frontier while keeping the three objectives visible.

The constrained expected-improvement acquisition for a candidate d is

$$a(d) = EI(d) \prod_{i=1}^{m} \Phi\left(- g_i \frac{(d)}{\sigma_{g_i}}(d) \right)$$

where EI(d) is the expected improvement of J_λ under the surrogate posterior, g_i(d) ≤ 0 are m constraint functions such as timing-slack, power-density, or area-budget limits, σ_{g_i}(d) is the predictive standard deviation of constraint i, and Φ is the standard normal cumulative distribution function. The product approximates the probability of feasibility under independent constraint uncertainty; correlated constraints require a joint Gaussian or Monte Carlo treatment. The acquisition function balances improvement against feasibility, which is critical in advanced-node flows where an infeasible timing or power-density point is not merely suboptimal but unusable.

The exploration loop proceeds in three stages. First, the current design and a small set of analytically generated variants initialize the surrogate. Second, the acquisition function selects a batch of candidate configurations, and the expensive flow evaluates them. Third, the surrogate is retrained on the augmented data, and λ or the constraint set is updated if the designer wishes to move along the trade-off surface. Active learning is relevant here because the most informative configurations are not necessarily the ones with the best predicted objective; they are the ones that reduce uncertainty in regions where the acquisition function is high [21]. High-dimensional Gaussian-process modeling is also relevant as a baseline, but its limitations motivate the graph-structured surrogate [22].

The framework differs from NSGA-II in its use of a learned surrogate and its constraint-aware acquisition. NSGA-II evaluates a population directly and can explore nonconvex frontiers, but it requires many expensive flow evaluations. The present framework is designed to reduce evaluations by querying where the surrogate predicts either high improvement or high information gain. The trade-off is that surrogate error can mislead the search, so trust-region control and periodic re-evaluation of the incumbent are necessary. The framework also differs from pure reinforcement learning because it does not learn a sequential policy over placement or routing actions; it treats design space exploration as a constrained optimization problem over a mixed configuration space, with the graph neural network acting as a surrogate rather than as an end-to-end policy.

5. Methodological Framework and Algorithmic Workflow

The methodological workflow begins with graph construction. For each candidate configuration, the circuit netlist is parsed into cell, net, terminal, and macro nodes. Connectivity and hierarchy edges are inserted, and node and edge features are computed from

the library and the current design state. Design variables are attached to the relevant nodes or to the graph-level attribute vector. Normalization is performed per feature using statistics from the initial design set. The graph construction step must be deterministic so that repeated evaluations of the same configuration produce the same graph; nondeterministic graph construction would introduce label noise and undermine both surrogate training and acquisition.

The second step is surrogate initialization and training. A small warm-start set is generated by analytical models, previous design iterations, or expert-chosen variants. The graph neural network is trained with the Gaussian negative log-likelihood objective, using a held-out split for early stopping and uncertainty calibration. Calibration is important because the acquisition function uses both the mean and the variance. A surrogate that is accurate on average but overconfident in unexplored regions can cause the optimizer to exploit false improvements. Temperature scaling or conformal calibration can be applied to the variance outputs if needed, but the calibration procedure must be specified before the exploration loop begins.

The third step is constrained acquisition. For each λ setting, the scalarized objective is computed from the surrogate mean, and the acquisition score is computed from expected improvement and feasibility probability. Candidate configurations are generated by perturbing continuous variables and by swapping discrete variables subject to legality rules. Legality rules include minimum spacing, maximum slew, maximum capacitance, and library compatibility. The acquisition function is evaluated on the legal candidates, and a batch is selected with diversity promotion to avoid redundant queries. The batch size is a design choice that trades parallel evaluation throughput against surrogate update frequency.

The fourth step is evaluation and update. The selected configurations are passed to the signoff or high-fidelity analysis flow, and the resulting power, frequency, and area values are added to the training set. If a configuration violates a hard constraint, it is labeled as infeasible and used to train the constraint heads. The surrogate is then retrained, and the incumbent design is updated if a feasible configuration improves the scalarized objective. The loop terminates when the evaluation budget is exhausted, when the predicted improvement falls below a tolerance, or when the Pareto frontier stabilizes. The termination criterion should be defined in terms of the actual design objective, not only in terms of surrogate loss.

The workflow is intentionally modular. The graph construction, message-passing architecture, predictive heads, and acquisition function are separable. This modularity allows a design team to replace the surrogate with a different graph model or to replace the acquisition rule with a Pareto-based selection strategy without rewriting the entire flow. It also allows the framework to be used with different constraint sets, such as timing-driven exploration, power-driven exploration, or area-constrained exploration. The modularity is not a claim of universal superiority; it is a practical requirement for adopting a learned surrogate in an engineering flow where the signoff environment, library, and process node may change.

6. Technical Discussion: Trade-offs, Uncertainty, and Boundary Conditions

The power-performance-area trade-off is physically grounded. Increasing supply voltage or upsizing gates can improve frequency but increases dynamic power and area. Reducing area by using smaller cells can increase delay and leakage variability. Buffer insertion can improve timing but adds area and switching power. The graph neural network surrogate can learn these relationships because the message-passing operators see the local cells, nets, and

path structure that mediate them. However, the surrogate is only as good as its training distribution. If the exploration loop proposes configurations far from the training set, the graph embedding may be extrapolating, and the predictive variance may not be reliable. Trust-region constraints on continuous variables and legality constraints on discrete variables reduce this risk but do not eliminate it.

Uncertainty handling is central. The acquisition function uses variance to balance exploration and exploitation. If the variance is underestimated, the optimizer may repeatedly query near the current incumbent and miss better regions. If the variance is overestimated, the optimizer may waste evaluations on uninformative regions. Calibration on a held-out set and periodic re-evaluation of the incumbent are therefore necessary. The constrained acquisition function also assumes independence among constraint uncertainties; when timing, power density, and area constraints are correlated, the product of feasibility probabilities can be biased. A joint Gaussian approximation or Monte Carlo sampling from the surrogate posterior is more appropriate when the constraint heads are strongly correlated.

Graph scaling is a second boundary condition. Advanced VLSI graphs can contain millions of nodes. Full-batch message passing is not feasible for large designs, so neighborhood sampling or hierarchical graph coarsening is required. Sampling introduces variance in the embedding and can obscure long-path effects. Hierarchical coarsening can preserve global structure but may lose local timing detail. The choice between sampling and coarsening depends on the target: if the objective is early-stage exploration, coarsening may be sufficient; if the objective is late-stage optimization, local path detail is more important and sampling with path-aware neighborhoods is preferable.

Surrogate exploitation is a third failure mode. Bayesian optimization can exploit regions where the surrogate predicts a large improvement even if the true flow would reject the configuration. This is not a defect of the acquisition function alone; it is a consequence of optimizing a learned model. The remedy is to maintain a trust region around evaluated configurations, to include a diversity term in batch selection, and to verify the incumbent with the high-fidelity flow at regular intervals. The framework should not be used as a replacement for signoff analysis; it is a query-selection mechanism that reduces the number of signoff evaluations.

Finally, the framework's scientific value depends on transparent evaluation. A rigorous assessment would compare the graph neural network surrogate with tree ensembles, Gaussian processes, and analytical models on the same design space, using metrics such as rank correlation on the Pareto frontier, constraint-violation rate, and evaluations to reach a target improvement. Those metrics are not asserted here because they depend on the process node, library, and flow. The methodological contribution is the formalization of a topology-aware surrogate and its integration with constrained acquisition. The framework is most defensible when its assumptions are explicit: the graph features must capture the relevant physical effects, the training set must cover the exploration region, and the uncertainty estimates must be calibrated before they are used to guide expensive evaluations.

7. Conclusion

This manuscript has developed a graph neural network-guided design space exploration framework for power, performance, and area optimization in advanced VLSI circuits. The framework represents each candidate configuration as a heterogeneous circuit graph, learns message-passing embeddings of cells, nets, and design variables, and predicts power,

frequency, and area with a probabilistic multi-task surrogate. It integrates the surrogate with constrained Bayesian optimization through a normalized scalarization and a feasibility-weighted expected-improvement acquisition rule. The methodological analysis clarifies why graph structure is an appropriate inductive bias for advanced-node design, how the surrogate should be trained and calibrated, and which failure modes must be controlled during exploration. The framework does not collapse the three objectives prematurely; it keeps trade-offs visible through scalarization weights and Pareto exploration. Its practical significance lies in reducing expensive signoff evaluations while preserving topology-aware reasoning about timing, power, and area. The main technical challenges are distribution shift, graph scaling, constraint correlation, and surrogate exploitation. Addressing these challenges requires careful graph construction, calibrated uncertainty, trust-region control, and transparent evaluation against credible baselines.

References

1. Kahng, A. B. (2018). Machine learning applications in physical design: Recent results and directions. In Proceedings of the 2018 International Symposium on Physical Design (pp. 68–73). ACM.

2. Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J., Songhori, E., Wang, S., Lee, Y., Johnson, E., Pathak, O., Nazi, A., Pak, J., Tong, A., Srinivasa, K., Hang, W., Hui, E., Bansal, A., Cousins, D., Shafi, W., Kaufman, M., ... Dean, J. (2021). A graph placement methodology for fast chip design. Nature, 594(7862), 207–212.

3. Huang, G., Hu, J., He, Y., Liu, J., Ma, M., Shen, Z., Wu, J., Xu, Y., Zhang, H., Zhong, K., Xue, N., Tong, Y., Wang, P., Zhou, Y., Li, Y., & Xie, Y. (2021). Machine learning for electronic design automation: A survey. ACM Transactions on Design Automation of Electronic Systems, 26(5), 1–46.

4. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4–24.

5. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations.

6. Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks? In International Conference on Learning Representations.

7. Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2016). Gated graph sequence neural networks. In International Conference on Learning Representations.

8. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning (pp. 1263–1272). PMLR.

9. Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), 182–197.

10. Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., & de Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. Proceedings of the IEEE, 104(1), 148–175.

11. Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In Advances in Neural Information Processing Systems (Vol. 25, pp. 2951–2959).

12. Forrester, A. I. J., Sóbester, A., & Keane, A. J. (2008). Engineering design via surrogate modelling: A practical guide. Wiley.

13. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). ACM.

14. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32.

15. Rasmussen, C. E., & Williams, C. K. I. (2006). Gaussian processes for machine learning. MIT Press.

16. Peng, Hongwu, et al. "Rrnet: Towards relu-reduced neural network for two-party computation based private inference." arXiv preprint arXiv:2302.02292 (2023).

17. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In International Conference on Learning Representations.

18. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning (pp. 448–456). PMLR.

19. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929–1958.

20. Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

21. Settles, B. (2009). Active learning literature survey (Computer Sciences Technical Report 1648). University of Wisconsin–Madison.

22. Binois, M., & Wycoff, N. (2022). A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. ACM Transactions on Evolutionary Learning and Optimization, 2(2), 1–52.