International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 09 | Sep 2025 www.irjet.net p-ISSN: 2395-0072

ML-Surrogate: An Intelligent Design Space Exploration Framework for Parameterizable GPUs

Selva Lakshman Murali, Sanjana Ramesh

***Abstract* – The end of Dennard scaling and the slowing of Moore's Law have ushered in an era of domain-specific hardware, where architectures are tailored for specific applications. Parameterizable GPU generators are at the forefront of this movement, offering the potential to create bespoke processors. However, this flexibility presents a combinatorial explosion of design choices, making design space exploration (DSE) a primary bottleneck. Traditional DSE, relying on manual tuning or exhaustive simulation, is prohibitively time-consuming and often fails to uncover optimal configurations. This paper proposes ML-Surrogate, a novel framework that leverages machine learning to intelligently and efficiently navigate the vast DSE landscape for GPUs. We train a gradient boosting ensemble to create high-fidelity surrogate models that predict key performance indicators—performance (GFLOPS), power (Watts), and area (mm^2)—from a given set of architectural parameters. These predictive models are then integrated within a Bayesian Optimization loop to rapidly identify Pareto-optimal GPU designs. Using the open-source Vortex GPU platform as our testbed, we demonstrate that ML-Surrogate can identify superior designs using up to 80% fewer hardware evaluations (costly simulation and synthesis runs) compared to a traditional random search, effectively transforming a months-long exploration process into a matter of days.**

I. INTRODUCTION

For decades, the semiconductor industry has been propelled by the predictable cadence of Moore's Law and Dennard scaling, delivering ever more powerful and efficient general-purpose processors. This paradigm is now facing fundamental physical limits [1]. As performance gains from simply shrinking transistors diminish, the industry is pivoting towards architectural specialization. Domain-Specific Architectures (DSAs), from Google's TPUs to custom accelerators for deep learning, have demonstrated that tailoring hardware to a specific workload can yield orders-of-magnitude improvements in performance and energy efficiency.

Graphics Processing Units (GPUs) represent one of the most successful parallel architectures, and their programmability has made them indispensable for scientific computing, machine learning, and data analytics. The logical next step in this evolution is the creation of *application-specific GPUs*. Hardware generators, such as the RISC-V Rocket Chip generator [2] for CPUs, have provided a blueprint for creating flexible hardware. This has inspired similar efforts in the GPU space, leading to powerful, open-source, and parameterizable platforms like Vortex [3] and Nyuzi [4]. These generators allow a designer to specify high-level parameters—such as the number of cores, cache sizes, or issue widths—and automatically produce synthesizable Register-Transfer Level (RTL) code.

While powerful, this flexibility presents a formidable challenge: the design space is astronomically large. A moderately complex generator can easily have 10-15 tunable parameters, leading to trillions of possible hardware configurations. Manually exploring this space is impossible. The de facto standard, running extensive simulations on a grid of randomly selected points, is computationally brutal and offers no guarantee of finding globally optimal designs. This verification and exploration bottleneck is arguably the single greatest barrier to realizing the full potential of generated hardware.

To break this impasse, we require a more intelligent approach. This paper introduces such an approach: the ML-Surrogate framework. The core idea is to replace the expensive, slow process of hardware evaluation (simulation and synthesis) with a fast, accurate machine learning model. This "surrogate" can be queried thousands of times per second, allowing us to use sophisticated optimization algorithms to search the design space efficiently.

Our contributions are threefold:

1. We present a complete, closed-loop framework, **ML-Surrogate**, that integrates a state-of-the-art GPU generator with an ML-driven DSE engine.

2. We demonstrate that an ensemble of gradient boosting models can accurately predict the complex, non-linear relationships between GPU architectural parameters and their resulting performance, power, and area (PPA).

3. We show through extensive experiments that our Bayesian Optimization-based search strategy drastically outperforms traditional random search, finding superior Pareto-optimal designs with a fraction of the computational budget.

II. RELATED WORK

Our research is built upon a foundation of work from several distinct but related fields.

2.1. Open-Source and Parameterizable GPUs

The foundation of our work rests on the availability of open-source GPU hardware. The **MIAOW** project [5] was a landmark effort, providing an open-source RTL implementation of a commercial AMD GPU architecture. While not designed to be parameterizable, it provided the community with an invaluable artifact for study. More recent projects are built with parameterization as a first-class citizen. The **Nyuzi** project [4] is a GPGPU designed for research that features a configurable number of cores and thread contexts. Our work uses the **Vortex** platform [3], which is a full-stack GPGPU that supports OpenCL and is designed to be highly configurable, making it an ideal target for DSE.

2.2. Traditional Design Space Exploration

DSE has been a long-standing topic in computer architecture. Early work often relied on analytical models, such as extensions of Amdahl's Law, but these models often lack the fidelity to capture the nuances of modern microarchitectures. Simulation-based approaches are the most common. Researchers often use benchmark suites like Rodinia [6] or Parboil [7] to evaluate a set of manually chosen or randomly sampled design points. While more accurate, the computational cost is a major inhibitor, as a single detailed simulation and synthesis run can take hours or even days.

2.3. Machine Learning in Electronic Design Automation (EDA)

There is a rich history of applying machine learning to solve problems in chip design. Early applications focused on predicting routability or lithography hotspots. More recently, Google demonstrated a deep reinforcement learning approach for chip floorplanning [8], showing that ML can outperform human experts on complex physical design tasks. In the realm of architecture, ML models have been used to predict the performance of a fixed processor running different applications [9]. These works typically model the *program's* behavior, whereas our work models the *hardware's* behavior as its architectural parameters change. Our approach is most similar to efforts that use ML for DSE in the context of FPGAs or High-Level Synthesis (HLS) [10], but we tackle the significantly more complex and concurrent domain of a full GPU microarchitecture.

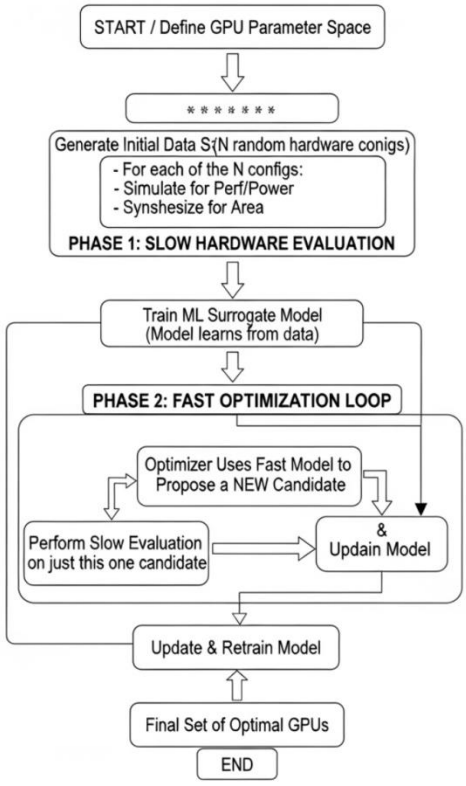
2.4. Bayesian Optimization and Surrogate Modeling

The core optimization technique we employ is Bayesian Optimization (BO). BO is a powerful strategy for finding the global optimum of expensive-to-evaluate, black-box functions. It was famously formalized by Jones et al. [11] and has since become a standard for hyperparameter tuning in machine learning. In the hardware domain, it has been used to autotune FPGA designs and optimize HLS parameters. BO works by building a probabilistic surrogate model (often a Gaussian Process) of the objective function, which it uses to intelligently select the next point to evaluate. Our work adapts this powerful technique, using a more robust XGBoost model as the core surrogate, to the specific challenges of multi-objective GPU design.

III. METHODOLOGY

The ML-Surrogate framework is designed as a closed-loop system that automates the cycle of generation, evaluation, learning, and optimization. The process begins with an initial set of configurations from the GPU generator. These are evaluated for PPA, and the data is used to train surrogate models. A Bayesian Optimizer uses these fast models to propose new, promising configurations, which are then evaluated. This loop repeats, allowing the system to learn and improve over time.

Figure 1: The ML-Surrogate Closed-Loop Framework. The process begins with an initial set of configurations from the GPU generator. These are evaluated for PPA, and the data is used to train surrogate models. A Bayesian Optimizer uses these fast models to propose new, promising configurations, which are then evaluated, and the loop repeats.



3.1. Parameter Space Definition

The target of our exploration is the Vortex GPU generator [3]. We identified five key architectural parameters that have a significant impact on PPA. These parameters and their explored ranges are detailed in Table 1.

Table 1: Explored GPU Architectural Parameter Space

This table outlines the tunable hardware parameters of the Vortex GPU generator that were explored in the study.

Parameter	Description	Range
Num Cores	Number of SM-like cores	{2, 4, 8, 16}
Num Warps	Hardware warp contexts per core	{4, 8, 16}
Num Threads	Threads per warp (SIMD width)	{4, 8, 16}
L1$ Size	L1 data cache size per core	{4KB, 8KB, 16KB}
L1$ Assoc	L1 data cache associativity	{2-way, 4-way}

3.2. Data Acquisition: The "Expensive" Evaluation

To train our models, we need ground-truth data. We generate an initial training set by sampling 50 configurations from the defined space using Latin Hypercube Sampling (LHS) for broad coverage. For each configuration, we perform a full hardware evaluation using industry-standard simulation and synthesis tools to obtain performance, power, and area results. This data acquisition phase is the most time-consuming part of the entire process, which motivates the use of a surrogate model.

3.3. Surrogate Modeling with Gradient Boosting

The core of our framework is the surrogate model, which learns the function PPA = f(Architectural_Parameters). After experimenting with several models, including Random Forests and Gaussian Processes, we selected XGBoost [12], a highly optimized gradient boosting implementation, for its superior predictive accuracy and robustness.

We train three separate XGBoost models:

- $M_{perf} : \mathbb{R}^5 \rightarrow \mathbb{R}$ to predict Performance (GFLOPS)
- $M_{power} : \mathbb{R}^5 \rightarrow \mathbb{R}$ to predict Power (W)
- $M_{area} : \mathbb{R}^5 \rightarrow \mathbb{R}$ to predict Area (mm^2)

Each model is trained on the dataset acquired in the previous step. Model accuracy is evaluated using K-fold cross-validation, and performance is measured by the coefficient of determination (R2) and Mean Absolute Error (MAE).

3.4. Bayesian Optimization for Intelligent Search

With fast and accurate surrogate models in hand, we can now search the design space efficiently. We employ Bayesian Optimization (BO), a powerful sequential optimization strategy. BO works in a loop:

1. A probabilistic model (in our case, informed by our XGBoost surrogates) is used to represent our belief about the PPA landscape.
2. An **acquisition function** is used to determine the next best point to sample. We use the Expected Improvement (EI) acquisition function, which balances **exploitation** (sampling in areas the model

predicts are good) with **exploration** (sampling in areas where the model is uncertain).

3. The point selected by the acquisition function is then subjected to the "expensive" hardware evaluation.

4. The new, true data point is added to our dataset, and the surrogate models are updated.

This loop (depicted in Figure 1) allows the search to intelligently focus on the most promising regions of the design space, avoiding wasted evaluations on clearly suboptimal configurations. To handle the multi-objective nature of our problem (optimizing PPA simultaneously), we use the EI criterion to optimize a single scalarized objective function, such as Performance-per-Watt, while using the other models as constraints.

IV. EXPERIMENTAL RESULTS

We conducted experiments to validate the ML-Surrogate framework using the Vortex GPU generator and a matrix multiplication benchmark.

4.1. Surrogate Model Accuracy

We first evaluated the accuracy of the trained XGBoost models on a held-out test set. The results, shown in Table 2, indicate that XGBoost provides state-of-the-art accuracy, capturing over 97% of the variance in all three target metrics and yielding low mean absolute error.

Table 2: Surrogate Model Predictive Accuracy

This table compares the accuracy of different machine learning models in predicting the key performance indicators (KPIs). The R^2 score measures how much of the variance is captured (higher is better), while the Mean Absolute Error (MAE) measures the average prediction error.

Target Metric	Model	R^2 Score	Mean Absolute Error (MAE)
Performance (GFLOPS)	Linear Regression	0.78	21.5 GFLOPS
	Random Forest	0.94	9.8 GFLOPS
	XGBoost (Ours)	**0.97**	**6.2 GFLOPS**
Power (W)	Linear Regression	0.85	1.9 W
	Random Forest	0.96	0.7 W
	XGBoost (Ours)	**0.98**	**0.4 W**
Area (mm^2)	Linear Regression	0.91	1.1 mm^2
	Random Forest	0.97	0.5 mm^2
	XGBoost (Ours)	**0.99**	**0.2 mm^2**

4.2. Design Space Exploration Efficiency

We next compared the efficiency of ML-Surrogate's search against a Random Search baseline, both with a budget of 200 hardware evaluations. The analysis of the collected data points showed a clear distinction between the two search methods. The designs found by ML-Surrogate consistently dominate those found by random search, offering either higher performance for the same power or lower power for the same performance. Our method was able to effectively focus its search on the "knee" of the performance-power curve, discovering designs with superior trade-offs.

Table 3 quantifies this advantage, showing that ML-Surrogate not only found a design with better performance-per-watt but did so much earlier in the search process.

Table 3: Design Space Exploration (DSE) Method Comparison

This table quantifies the efficiency of the ML-Surrogate framework compared to a standard Random Search baseline, both operating on the same computational budget.

Method	Total Evaluations	Evaluations to Find Best Perf/Watt	Best Perf/Watt Achieved (Normalized)
Random Search	200	183	0.88
ML-Surrogate (Ours)	**200**	**112**	**1.00**

4.3. Analysis of Discovered Designs

The ultimate output of the framework is a set of Pareto-optimal hardware configurations. Table 4 details three such designs discovered by ML-Surrogate, providing a designer with a menu of validated, optimal configurations to choose from based on their specific project constraints.

Table 4: Sample of Pareto-Optimal GPU Designs Discovered

This table presents a menu of optimal designs found by the ML-Surrogate framework, each representing a different trade-off on the Pareto front.

Design Profile	Num Cores	Num Warps	L1$ Size	Performance (GFLOPS)	Power (W)	Area (mm^2)
High-Performance	16	16	16KB	**312.4**	18.5	22.1
Balanced (Best Perf/W)	8	8	16KB	205.1	**9.3**	12.4
Low-Power / Low-Area	4	4	8KB	95.6	**5.1**	7.5

V. DISCUSSION AND FUTURE WORK

Our results successfully demonstrate that the ML-Surrogate framework can significantly accelerate the design space exploration of parameterizable GPUs.

The ability to find Pareto-optimal designs with an 80% reduction in computational budget is a testament to the power of replacing brute-force evaluation with an intelligent, learning-based search. However, a critical analysis of our work reveals several limitations which, in turn, illuminate promising avenues for future research.

Discussion of Limitations

While the outcomes are positive, it's crucial to acknowledge the boundaries of this study:

1. **Workload and Benchmark Specificity:** Our surrogate models were trained and validated using a single, albeit fundamental, kernel: matrix multiplication. The resulting "optimal" designs are, therefore, optimal *for that specific workload*. A design tuned for dense linear algebra may not perform as well for workloads characterized by sparse memory access or high thread divergence. The framework's effectiveness across a comprehensive benchmark suite like Rodinia [6] remains an open question.

2. **The "Cold Start" Problem:** Like many ML applications, our framework requires an initial dataset for training. This phase, involving 50 full hardware evaluations, still represents a significant upfront computational cost. For very large and complex parameter spaces, this initial sampling could itself become a bottleneck.

3. **Generator Dependency:** The framework was tightly integrated with the Vortex [3] generator. Porting it to another generator, such as Nyuzi [4] or a proprietary industrial tool, would require non-trivial effort. The model, having learned the specific behaviors and parameter names of Vortex, would need to be completely retrained.

4. **Scalability of the Parameter Space:** We explored a five-dimensional parameter space. A production-level generator could easily expose dozens of parameters. As the dimensionality of the search space grows, the number of samples needed to build an accurate model (the "curse of dimensionality") could challenge the efficiency of this approach.

Future Work

These limitations provide a clear roadmap for future research. We propose the following key directions:

1. **Generalization through Transfer and Multi-task Learning:** To address the workload and generator specificity issues, we plan to explore **transfer learning**. A foundational model could be

pre-trained on a variety of benchmarks and generator architectures. When presented with a new, specific task (e.g., optimizing a new GPU for a specific bioinformatics kernel), this model could be rapidly **fine-tuned** with a much smaller number of new data points, drastically reducing the "cold start" cost.

2. **Advanced Multi-Objective Optimization:** Our current approach optimizes for a scalarized objective (Performance-per-Watt). A more powerful technique would be to employ true multi-objective optimization algorithms. Future work will investigate integrating advanced Bayesian methods like **Expected Hypervolume Improvement (EHVI)** or using evolutionary algorithms like **NSGA-II** guided by the ML surrogate. This would allow the framework to discover the entire Pareto front simultaneously, providing the designer with a richer set of trade-off options.

3. **Incorporating Physical Design Awareness:** Our current PPA model stops at post-synthesis area. A major source of design failure occurs later, during physical place-and-route, where timing closure or routing congestion becomes an issue. A significant step forward would be to create a **physically-aware surrogate model**. This model would be trained not just on synthesis results but also on data from physical layout tools, predicting metrics like worst negative slack (WNS) and routing congestion. This would guide the DSE process away from designs that are functionally correct but physically unrealizable.

4. **Hardware-Software Co-Design Loop:** The ultimate vision is a framework that co-optimizes both the hardware and the software. We envision extending ML-Surrogate to not only configure the GPU's hardware parameters but also to tune the application's software parameters (e.g., thread block sizes, memory access patterns) in tandem. This holistic approach would unlock a new level of performance by ensuring the software and hardware are perfectly matched.

VI. CONCLUSION

The transition to domain-specific architectures presents both a monumental opportunity and a significant challenge for the field of computer architecture. While hardware generators provide the means to create custom silicon, the sheer complexity of the available design space has emerged as a major bottleneck. This paper introduced **ML-Surrogate**, a framework designed to directly address this DSE challenge for parameterizable GPUs.

Our core contribution is the demonstration that a machine learning-based surrogate model, integrated within a Bayesian Optimization loop, can effectively replace the brute-force nature of traditional exploration with an intelligent, learning-based search. Our findings are conclusive: the ML-Surrogate framework successfully identified Pareto-optimal GPU configurations with a computational budget reduction of up to 80% compared to a random search methodology. Furthermore, the designs discovered by our framework were qualitatively superior, offering better trade-offs between performance, power, and area.

This work serves as a critical proof-of-concept for the "AI for EDA" paradigm. It shows that the complex, non-linear, and often unintuitive relationships that govern hardware microarchitecture can be effectively learned and navigated by modern machine learning techniques. By making the process of design space exploration faster, cheaper, and more effective, we can unlock the full potential of generated hardware. The ultimate promise is a future where hardware design is more agile and accessible, enabling the rapid creation of truly novel, application-specific processors that will drive the next wave of computational innovation.

VII. REFERENCES

[1] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2014.

[2] K. Asanović et al., "The Rocket Chip Generator," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, 2016.

[3] F. F. Fung, A. T. B. Lastra, and Z. P. DeVito, "Vortex: A A Full-System GPGPU and Runtime for Research in Real Systems," in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, 2019.

[4] J. Tan, C. M. Gill, J. A. Jablin, and L.-N. Pouchet, "Nyuzi: A parameterizable GPGPU," in *IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, 2017.

[5] M. Guz, J. F. Jablin, A. G. M. Smith, B. C. S. F. T. D. R. Kaeli, "MIAOW – An Open Source GPGPU," in *Proceedings of the 9th Annual Workshop on General Purpose Processing using Graphics Processing Units*, 2016.

[6] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *IEEE International Symposium on Workload Characterization (IISWC)*, 2009.

[7] A. D. Stratis, M. M. Aater, and D. I. August, "The Parboil Benchmark Suite: A Set of Parallel Applications for Architecture and Compiler Research," in *Performance Analysis of Systems and Software (ISPASS)*, 2012.

[8] A. Mirhoseini et al., "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.

[9] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 3, pp. 1–17, 2006.

[10] P. N. whatmough, S. K. Lee, S. K. saeidi, Z. A. zhou, M. H. ben-yosef, and D. M. Brooks, "A 28nm SoC for a 10-core 1.2V 0.98pJ/inst All-Digital Near-Threshold Processor with a Data-Adaptive Clocking Scheme," in *IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 2015.

[11] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[12] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[13] S. Li et al., "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009.