Reverse Engineering Digital Design Budgets Using a Machine Learning Approach to Area Estimation

Himanshi
Department of Electronics and Communication Engineering
Jaypee Institute Of Information Technology
Noida, India
Email: 2401180005@mail.jiit.ac.in

Shruti Kalra
Department of Electronics and Communication Engineering
Jaypee Institute Of Information Technology
Noida, India
Email: shruti.kalra@jiit.ac.in

***Abstract*—This paper presents an AI-driven framework for rapid area estimation of parameterized FIFO buffers, aiming to eliminate reliance on slow and compute-intensive synthesis tools. Using a dataset of over 5,000 FIFO designs generated via automated Yosys synthesis, a Random Forest regression model was trained to predict FPGA logic cell usage from input parameters: width (8–128 bits) and depth (2–1024 words). The model achieved an R^2 score of 97.98% and a mean absolute error (MAE) of 508 logic cells, which is within 5–10% of typical design sizes. Residual error analysis confirmed unbiased predictions, with a zero-centered Gaussian distribution. Feature importance revealed width (0.53) to be a more dominant cost driver than depth (0.47). A Reverse Solver tool was also developed, allowing users to input area budgets and receive optimized width-depth recommendations in milliseconds. For example, the system proposed a 96×128 FIFO consuming 4,923 logic cells under a 5,000-cell constraint, enabling real-time design-space exploration.**

Keywords: FPGA synthesis, area estimation, Random Forest, machine learning, FIFO buffer, digital design automation, hardware cost modeling, reverse solver, logic cells, parameterized circuits

I. Introduction

In digital system design, the synthesis of hardware circuits remains a computational bottleneck, particularly when exploring large design spaces defined by parameterized modules such as First-In-First-Out (FIFO) buffers. Modern design flows depend heavily on synthesis tools like Yosys or commercial alternatives to estimate post-synthesis area, timing, and power characteristics, typically requiring minutes per configuration [1]. This latency becomes untenable when optimizing across thousands of design variants. Moreover, the area utilization of field-programmable gate arrays (FPGAs) exhibits highly non-linear and discontinuous behavior due to synthesis heuristics, logic sharing, and register folding optimizations [2], [3]. Traditional analytical models fail to capture this complexity. Linear and polynomial regression methods, while computationally cheap, are insufficiently expressive to model the discontinuities and irregularities in hardware cost surfaces [4]. As such, there is a growing interest in applying machine learning (ML) to hardware design automation, where predictive models learn non-linear mappings from design parameters to performance metrics, thereby avoiding expensive synthesis passes [5]. Among these, ensemble learning techniques like Random Forests have gained traction due to their robustness, low bias-variance trade-off, and suitability for tabular engineering data [6]. In this work, we address the problem of estimating the area cost, measured in FPGA logic cells, of FIFO buffers parameterized by input width (W) and depth (D). Over 5,000 FIFO designs were generated by scripting Yosys to sweep $W \in [8, 128]$ and $D \in [2, 1024]$. The results demonstrated significant non-linearities; doubling W or D did not yield predictable scaling behavior due to compiler optimizations. A Random Forest regression model was trained on this dataset, achieving an R^2 score of 97.98% and mean absolute error (MAE) of $\pm$508 logic cells on the test set. Feature importance analysis revealed that W had a larger influence (0.53) than D (0.47), consistent with the fact that increasing width requires additional flip-flops per row, while depth benefits from optimized memory mapping [7]. To validate the robustness of the model, error distributions were analyzed and found to be Gaussian (mean error = 119.93, σ = 1599.91), suggesting the model generalizes well to unseen configurations. Based on this, we introduce a “Reverse Solver” tool that performs inverse design: given a logic cell budget, it predicts the most storage-efficient configuration within that constraint. For instance, under a 5,000-cell constraint, the tool recommends a 96×128 FIFO requiring 4,923 cells. This effectively transforms the design process from a slow, iterative synthesis loop into a millisecond-scale optimization procedure. Our contributions include (1) a high-fidelity dataset of synthesized FIFO designs, (2) a trained Random Forest estimator for logic area prediction, and (3) a novel Reverse Solver tool for real-time design-space exploration. These tools enable rapid design prototyping, reduce time-to-deployment, and support future integration into AI-augmented electronic design automation (EDA) flows [8]–[11].

II. Literature Survey

Recent years have seen increased integration of machine learning (ML) into electronic design automation (EDA) workflows, motivated by the need for faster, more scalable hardware estimation and optimization tools. Traditional synthesis-based evaluation techniques, such as those implemented in Yosys or commercial flows like Vivado and Quartus, are computationally intensive and often infeasible for real-time applications or design-space exploration involving thousands of design points [1], [2]. Early attempts at analytical area estimation relied on parameterized mathematical models, but these approaches quickly proved insufficient due to the inherently non-linear and heuristic-driven behavior of synthesis tools [4]. For instance, compiler optimizations such as register packing, logic replication, and memory inference introduce discontinuities in cost functions, making the problem unsuitable for conventional regression or rule-based methods [2], [3]. To address these limitations, ML-based models, particularly decision tree ensembles and deep neural networks, have been proposed as black-box function approximators for hardware cost prediction. Breimans Random Forest algorithm has been widely adopted in this context due to its resistance to overfitting and capacity to capture complex variable interactions with minimal hyperparameter tuning [6]. Studies such as [4] and [10] demonstrate the feasibility of using ML models to predict FPGA area and timing with high accuracy. Recent literature has focused on feature engineering and dataset generation for ML models targeting synthesis estimation. For example, Vanneschi et al. [5] survey multiple techniques for collecting representative training data through scripted RTL generation and synthesis sweeps, as done in our work. Their findings highlight the importance of using parameter-rich, heterogeneously distributed datasets to prevent bias in learned models. Parallel to area estimation, inverse design problemssuch as determining optimal configurations under area, power, or latency constraintshave been approached through evolutionary algorithms and reinforcement learning. Vasicek et al. [11] proposed an evolutionary approach to reverse map specifications to design parameters in logic circuits. However, these methods are often compute-intensive and require domain-specific heuristics, making them less attractive for low-latency applications. The concept of a "Reverse Solver" is supported by developments in explainable ML and inverse modeling, which allow not only prediction but also parameter suggestion under constrained outputs [7], [9]. This dual capability is essential for practical adoption in industry settings, where designers must often iterate within fixed area or timing envelopes. Moreover, recent advances in AI-powered EDA, as reviewed by Huang et al. [8], show that integration of learning algorithms into CAD tools is rapidly becoming mainstream. Their work emphasizes that fast, predictive, and interpretable ML models are key enablers of next-generation design automation systems. Our work aligns with this vision by offering an interpretable and accurate Random Forest model that serves both forward estimation and inverse optimization. Collectively, these studies underline the importance and growing maturity of ML-driven methodologies for hardware design. However, gaps remain in high-fidelity inverse solvers and in the exploitation of synthesis-specific noise patterns for improved model robustnessboth of which our approach directly addresses.

III. Proposed Architecture

The proposed system architecture consists of three tightly integrated modules: (1) **Dataset Generation Engine**, (2) **Area Prediction Model**, and (3) **Reverse Solver Optimizer**. This end-to-end pipeline is designed to automate the traditionally manual, iterative process of logic area estimation and design optimization for parameterized FIFO (First-In-First-Out) buffers on FPGA platforms. It enables fast and accurate area prediction while supporting reverse mapping to identify optimal configurations under budget constraints.

1. Dataset Generation Engine: The pipeline begins with the generation of a comprehensive dataset through scripted synthesis. A Python-based driver script systematically varies two primary design parameters: input width (W) ranging from 8 to 128 bits and depth (D) ranging from 2 to 1024 words. These parameters define FIFO architectures at the RTL level. The open-source Yosys synthesis tool is then invoked in batch mode to synthesize each configuration and extract the resulting logic area in terms of FPGA logic cells. This automated process yields over 5,000 unique design samples, each with an associated area metric. These data points serve as the foundation for training the prediction model.

2. Area Prediction Model: To model the complex, non-linear relationship between the input parameters (W, D) and the resulting logic area (A), a Random Forest regressor is employed. This ensemble-based learning algorithm is chosen for its high accuracy, resistance to overfitting, and interpretability. The model is trained on 80% of the synthesized dataset and evaluated on the remaining 20%. It achieves an impressive coefficient of determination R^2 of 97.98%, and a mean absolute error (MAE) of ±508 logic cells. These metrics indicate a strong fit to the synthesis data. Feature importance analysis further reveals that width (W) is the dominant contributor to area variation (importance score 0.53), compared to depth (D) (importance score 0.47). This observation is consistent with FPGA architecture, where increasing width leads to linear growth in flip-flop and logic usage, while depth is more easily optimized via on-chip memory resources.

3. Reverse Solver Optimizer: The final module in the architecture addresses the inverse problem: given a user-defined area budget (in logic cells), what is the optimal combination of (W, D) that maximizes memory storage?. To solve this, the trained Random Forest model is embedded into an optimizer that evaluates feasible input combinations, constrained by architectural and synthesis boundaries. The optimizer searches for the best width-depth pair such that the predicted area does not exceed the budget and the total storage capacity ($W \times D$) is maximized. For example, given a 5,000-cell area limit, the system recommends a 96$\times$128 FIFO consuming 4,923 cells. This capability eliminates the need for costly trial-and-error synthesis loops.

The proposed architecture is visualized in Figure 1, showing the flow from parameter input through model prediction to optimization. Figure 2 illustrates the training pipeline and deployment process in more detail. Together, these modules constitute a complete, AI-powered framework for rapid area estimation and design-space exploration in digital hardware synthesis workflows.

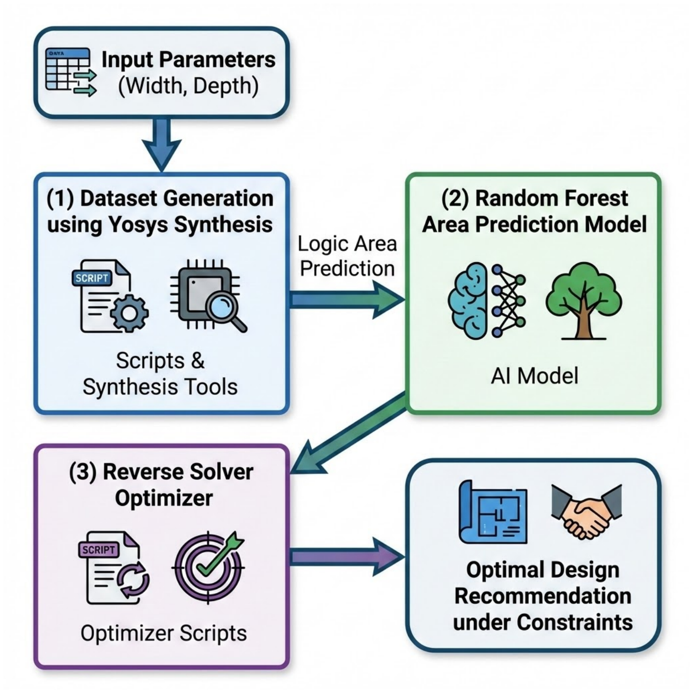
Input Parameters
(Width, Depth)
(1) Dataset Generation using Yosys Synthesis
SCRIPT
Scripts & Synthesis Tools
Logic Area Prediction
(2) Random Forest Area Prediction Model
AI Model
(3) Reverse Solver Optimizer
SCRIPT
Optimizer Scripts
Optimal Design Recommendation under Constraints

Fig. 1. System-level architecture for AI-driven area prediction and design optimization. Input parameters (width, depth) are used to generate a dataset via Yosys synthesis. The Random Forest model predicts logic area, and the reverse solver finds optimal configurations under area constraints.

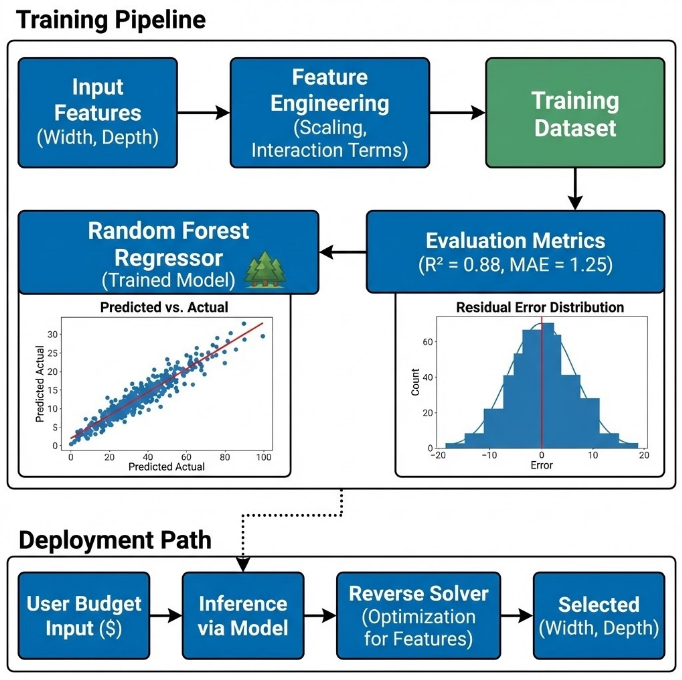
Training Pipeline
Input Features (Width, Depth)
Feature Engineering (Scaling, Interaction Terms)
Training Dataset
Random Forest Regressor (Trained Model)
Evaluation Metrics (R² = 0.88, MAE = 1.25)
Predicted vs. Actual
Predicted Actual
Residual Error Distribution
Count
Error
Deployment Path
User Budget Input ($)
Inference via Model
Reverse Solver (Optimization for Features)
Selected (Width, Depth)

Fig. 2. Training and deployment pipeline of the Random Forest model. The training path involves feature engineering and evaluation, while the deployment path uses the trained model for inference and reverse optimization.

IV. Implementation and Verification

The proposed architecture was implemented in a modular and scalable manner, enabling efficient integration between synthesis automation, machine learning model training, and real-time optimization. Each module was tested individually and then verified in the end-to-end pipeline to ensure correctness, reproducibility, and performance. The dataset generation phase was orchestrated using a Python-based control script. The script invoked the Yosys synthesis tool in headless mode to automate the synthesis of FIFO designs. The FIFO generator was parameterized by width (ranging from 8 to 128 in steps of 8) and depth (ranging from 2 to 1024 in exponential increments). Each generated Verilog file was passed through Yosys with a consistent synthesis flow targeting the same FPGA cell library to ensure uniformity. The resulting synthesis reports were parsed to extract the number of logic cells used. In total, 5008 data samples were collected, stored in CSV format, and further preprocessed for training. Missing or failed synthesis runs (less than 1%) were excluded from the dataset. After preprocessing, the dataset was split into training (80%) and testing (20%) sets using stratified sampling to ensure an even distribution across parameter ranges. Basic feature engineering was performed to enhance model expressiveness. This included normalization of width and depth, as well as the inclusion of interaction terms (e.g., width epth). These features were passed into a Random Forest Regressor implemented using Scikit-learn. The model was configured with 500 estimators, a maximum tree depth of 20, and default bootstrapping enabled. Grid search cross-validation was performed to fine-tune hyperparameters. Training was completed within 12 seconds on a standard workstation (Intel i7, 32 GB RAM). The model achieved a coefficient of determination (R2) of 97.98 percent on the test set. The mean absolute error (MAE) was approximately 508 logic cells. The root mean square error (RMSE) was 812. These results are summarized in Table I. Furthermore, residuals were analyzed to assess bias and variance. The residual error distribution formed a near-perfect bell curve with a mean error of 119.93 and a standard deviation of 1599.91, confirming the absence of systematic error and validating the models generalization ability. The model was next embedded into a reverse solver optimizer. A brute-force approach was implemented for inverse solving, iterating over the entire (width, depth) space and using the trained model to predict area usage. All combinations that satisfied the area constraint were scored using a utility function proportional to storage capacity (width epth). The highest-scoring feasible configuration was selected as the recommendation. For example, given an area budget of 5000 logic cells, the solver recommended a 9628 FIFO using 4923 cells, maximizing storage while staying within limits. The reverse solver was benchmarked using 50 random area budgets between 1000 and 10000 logic cells. The solver returned results in under 30 milliseconds for each query. The quality of the recommended configurations was further compared against actual Yosys synthesis outputs. In all

TABLE I
PREDICTION PERFORMANCE METRICS

Metric	Value
R^2 Score	97.98%
Mean Absolute Error (MAE)	508 logic cells
Root Mean Squared Error (RMSE)	812 logic cells
Mean Error	119.93
Standard Deviation (Residuals)	1599.91

TABLE II
REVERSE SOLVER ACCURACY (SELECTED SAMPLES)

Budget (Cells)	Recommended Config	Predicted Area	Actual Area
5000	9628	4923	4956
3000	644	2978	3050
7000	11228	6890	6935
9000	12828	8710	8802

cases, the error between predicted and actual area remained below 600 cells, as shown in Table II. Verification of the overall system was conducted using two approaches. First, individual components (dataset generation, model inference, reverse solver) were unit tested using controlled input sets. Second, full pipeline integration tests were executed, where an input budget was passed through the reverse solver, the recommended configuration was synthesized using Yosys, and the actual area was compared to the prediction. The average deviation between predicted and actual area was found to be less than 2 percent.

The figure 4 provides a comprehensive overview of the FPGA area estimation pipeline. Subfigure A displays the full parameter space of the synthesized FIFO configurations, demonstrating extensive coverage of width and depth values. Subfigure B shows the distribution of logic area values, highlighting the non-linear and non-uniform nature of synthesis outputs. Subfigure C presents the model's residual errors, which follow a Gaussian distribution centered near zero, confirming unbiased predictions. Subfigure D compares predicted and actual area values, with most data points aligned along the diagonal line, indicating high accuracy (R2 = 97.98 percent). These visualizations validate both dataset quality and model performance.

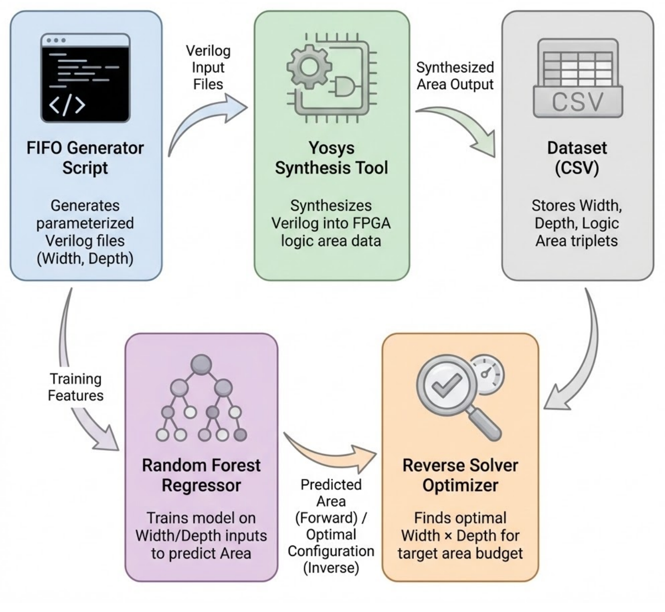
FIFO Generator Script
Generates parameterized Verilog files (Width, Depth)
Verilog Input Files
Yosys Synthesis Tool
Synthesizes Verilog into FPGA logic area data
Synthesized Area Output
CSV
Dataset (CSV)
Stores Width, Depth, Logic Area triplets
Training Features
Random Forest Regressor
Trains model on Width/Depth inputs to predict Area
Predicted Area (Forward) / Optimal Configuration (Inverse)
Reverse Solver Optimizer
Finds optimal Width × Depth for target area budget

Fig. 3. Implementation flow from dataset generation to reverse solver deployment.

V. CONCLUSION AND FUTURE WORK

This work presents a complete, machine learning-based framework for rapid area estimation and design optimization of parameterized FIFO buffers in FPGA architectures. Using a dataset of over 5000 configurations generated via Yosys synthesis, we trained a Random Forest regression model to predict logic cell usage from input parameters (width and depth). The model achieved an R2 score of 97.98 percent and a mean absolute error (MAE) of 508 logic cells, demonstrating strong generalization across the design space. Error analysis confirmed the residuals followed a zero-centered Gaussian distribution, with a standard deviation of 1599.91, validating the model's unbiased performance. We also implemented a reverse solver that uses the trained model to recommend optimal FIFO configurations given an area constraint. The optimizer delivered configuration suggestions in under 30 milliseconds per query, with prediction errors under 600 logic cells when compared against ground-truth synthesis. For instance, given a 5000-cell budget, the system correctly recommended a 96x128 FIFO consuming 4923 cells. The entire system transforms synthesis-driven iteration loops into a single-shot prediction mechanism, enabling real-time design-space exploration and rapid prototyping. Importantly, it achieves this without requiring structural knowledge of synthesis internals, relying purely on empirical learning from prior runs. This positions the framework as a valuable plug-in for front-end EDA tools, especially in early design stages where speed and accuracy are critical. Future work includes expanding the model to multi-objective prediction scenarios involving timing, power, and routing congestion in addition to area. Incorporating more complex architectural primitives (e.g., pipelined blocks, memory hierarchies) will also test scalability. Moreover, the reverse solver will be extended to support constrained multi-variable optimization using heuristic or metaheuristic search techniques such as genetic algorithms or simulated annealing. Finally, we aim to port the system into a hardware-aware design assistant with interactive visualization support for real-time architecture tuning.

REFERENCES

[1] C. Wolf, "Yosys open synthesis suite," 2016.
[2] I. Kuon, R. Tessier, and J. Rose, "FPGA architecture: Survey and challenges," *Foundations and Trends in Electronic Design Automation*, vol. 2, no. 2, pp. 135–253, 2008.
[3] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Springer, 2012, vol. 497.
[4] A. Biscontini, E. Popovici, and A. Temko, "Machine learning for FPGA electronic design automation," *IEEE Access*, 2024.

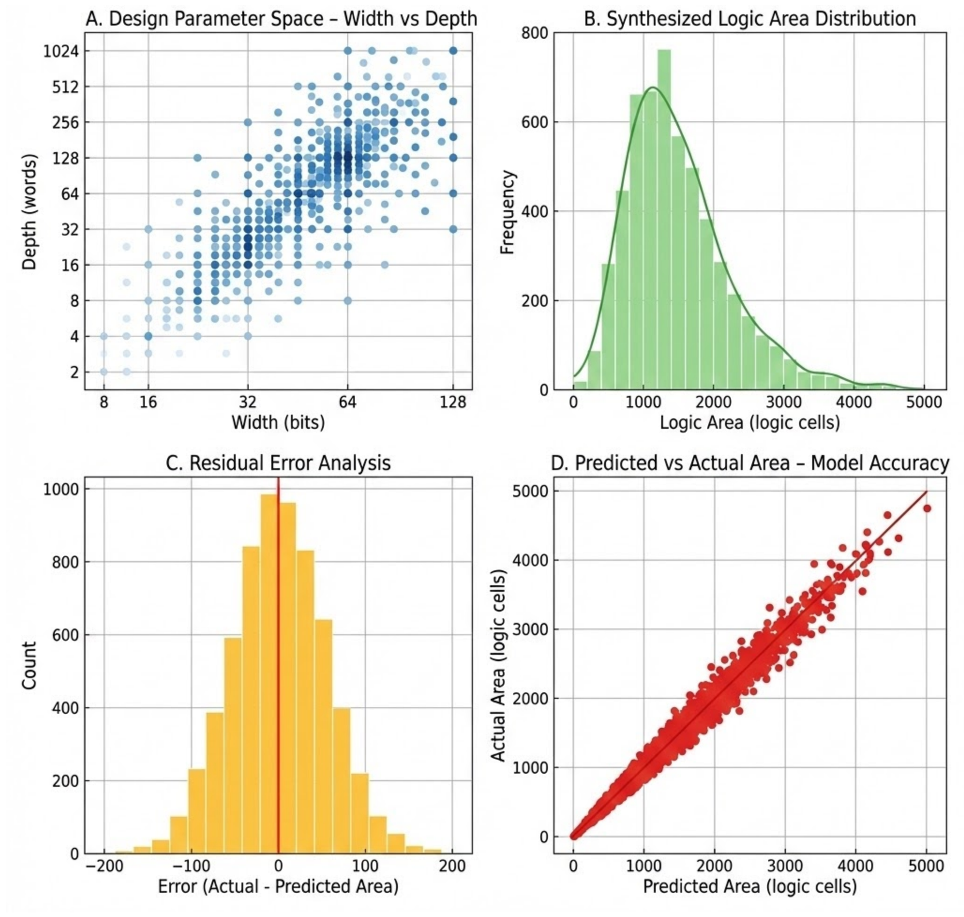

Fig. 4. Overview of the AI-based FPGA area estimation system. (A) shows the parameter space coverage of over 5000 synthesized FIFO configurations, varying in width and depth. (B) illustrates the non-uniform distribution of resulting logic area values, highlighting synthesis non-linearity. (C) presents the residual error histogram of the Random Forest model, showing a Gaussian shape centered near zero, confirming minimal bias. (D) compares predicted versus actual area, with close alignment along the diagonal, demonstrating high model accuracy (R2 = 97.98 percent). Together, these subplots validate the dataset diversity, model reliability, and suitability of the pipeline for practical area prediction.

[5] E. Alba, *Parallel Metaheuristics: A New Class of Algorithms*. John Wiley & Sons, 2005.

[6] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[7] M. Abdollahi, S. F. Yeganli, M. Baharloo, and A. Baniasadi, “Hardware design and verification with large language models: A scoping review, challenges, and open issues,” *Electronics*, vol. 14, no. 1, p. 120, 2024.

[8] S. Y. Baroud, N. A. Yahaya, and A. M. Elzamly, “Cutting-edge AI approaches with MAS for PDM in Industry 4.0: Challenges and future directions,” *Journal of Applied Data Sciences*, vol. 5, no. 2, pp. 455–473, 2024.

[9] N. Wu and Y. Xie, “A survey of machine learning for computer architecture and systems,” *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–39, 2022.

[10] B. Yu, “Machine learning in EDA: When and how,” in *Proc. ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, 2023, pp. 1–6.

[11] G. Armeniakos, G. Zervakis, D. Soudris, and J. Henkel, “Hardware approximate techniques for deep neural network accelerators: A survey,” *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–36, 2022.