POLITECNICO DI TORINO
Repository ISTITUZIONALE

Predicting Digital Layout Success in Analog-on-Top Designs Using Machine Learning

Original
Predicting Digital Layout Success in Analog-on-Top Designs Using Machine Learning / Daghero, F., Faraone, G., Serianni, E., Di Carolo, N., Licastro, D., Franchino, G.A., Grosso, M., Pagliari, D.J.. - ELETTRONICO. - (2025), pp. 60-64. (23rd IEEE Interregional NEWCAS Conference, NEWCAS 2025 Paris (FR) 22-25 June 2025) [10.1109/newcas64648.2025.11107059].

Availability:
This version is available at: 11583/3003249 since: 2025-09-22T15:20:02Z

Publisher:
Institute of Electrical and Electronics Engineers

Published
DOI:10.1109/newcas64648.2025.11107059

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
IEEE postprint/Author's Accepted Manuscript

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

31 October 2026

Predicting Digital Layout Success in Analog-on-Top Designs using Machine Learning

Francesco Daghero*, Gabriele Faraone†, Eugenio Serianni†, Nicola Di Carolo†, Dario Licastro†, Giovanna Antonella Franchino†, Michelangelo Grosso†, Daniele Jahier Pagliari*
*Politecnico di Torino, name.firstsurname@polito.it, †STMicroelectronics, name.surname@st.com

***Abstract*—Designing Analog-On-Top Mixed Signal (AMS) Integrated Circuits (ICs) is a labor-intensive and mainly manual process. A crucial step in this flow is reserving specific areas on the top-level layout to place digital blocks. This requires multiple time-consuming iterations between digital and analog teams, as several design characteristics are to be considered to determine whether the area is sufficient. Existing automated solutions are generally limited, as they often yield limited accuracy or have been tested only on simplistic use cases. In this work, we propose a Machine-Learning (ML) solution to determine whether the reserved area for a digital block is sufficient, handling this task as a supervised classification problem. Specifically, we perform an extensive benchmark of different ML models on labeled production-level designs, obtaining up to 94.8% F1 score. Finally, we provide an in-depth analysis of how different design features impact the prediction quality of several ML models, showing that a feature reduction technique can improve the final accuracy by up to 8.6%.**

***Index Terms*—EDA, Machine Learning, Place and Route, Analog-Mixed-Signal**

I. Introduction and Related Works

The Physical Design of Analog and Mixed-Signals (AMS) integrated circuits (ICs) commonly follows an "Analog-on-Top" approach. Digital subsystems are incorporated as pre-designed intellectual property (IP) blocks, which is particularly effective when analog circuitry constitutes most of the design. In the Analog-on-Top strategy, shown in Fig. 1, the placement and routing of analog components are prioritized by the top-level layout. This is generally done manually, as, even if time-consuming, it allows designers to optimize for variability control, noise reduction, and power efficiency [1], [2]. Digital elements are placed on the remaining available silicon area.

Once such area is defined, it requires validation by a team of digital designers who run the complete Place-and-Route (PnR) flow using digital Electronic Design Automation (EDA) tools. Timing, area, and power constraints can only be verified at the end of the PnR flow because, for example, while the placement of digital cells might be successful, geometric constraints may prevent achieving timing or power closure during later clock tree synthesis and routing steps. If constraints validation fails, the analog design team must take corrective actions such as altering the shape of the area, relocating input/output pins on the digital blocks, or re-routing signal or power supply nets in the analog top-level design. This often requires multiple iterations between the analog and digital teams, which can quickly become a bottleneck in the process.

To address this issue and enhance productivity in the AMS-IC physical design phase, an automated solution for determining the *layout feasibility* of digital blocks in a given area is highly desirable.

Other works have proposed various estimation techniques in the Digital PnR flow. In [3], [4] the authors introduce a probabilistic model to generate a range of estimates for feasible shapes of the block layout. The authors of [5] implement an equation to estimate the Configurable Logic Blocks (CLB) required by a design, and use it to determine if it fits inside a Field Programmable Gate Arrays (FPGA). More recently, Machine Learning (ML)-based approaches have also been explored. In [6], the authors obtain global and detailed routing statistics respectively with a MultiLayer Perceptron (MLP) and a Decision Tree (DT), using as input features such as the target clock period and the number of layers. However, no information about the training data is provided, leaving the generalization of this approach to complex layouts as an open point. The authors of [7], [8] use digital components specifications as input of an ML model to estimate their area requirements. They benchmark Random Forests (RFs), Adaptive Boosting, Support Vector Machines (SVMs), and MLPs, but limit their dataset to simple blocks, such as adders and multipliers. The work closest to ours is [9], where the authors benchmark a DT to estimate the layout feasibility of a design, achieving 76% balanced accuracy. However, they report high standard deviation, with folds achieving 50% accuracy. The model's limited predictive capabilities are due to preferring superior interpretability, thus considering only DT models. In contrast, our work introduces an automated ML-based solution that significantly outperforms previous methods in determining the feasibility of placing digital blocks within a given area. Specifically, our approach can predict, *before the digital PnR flow is started*, whether it will complete without overlaps and design rule check (DRC) errors, such as shorts or wire spacing issues, at the end of the detailed routing phase.

By providing such *PnR feasibility prediction*, our method aids analog layout designers during the top-level placement optimization phase of AMS ICs, reducing potential bottlenecks and leading to a more time- and resource-efficient design process. We obtain this prediction feeding ML models with high-level features of the input netlist, layout characteristics, and technology information We collect and label a dataset of real-world designs using Bipolar-CMOS-DMOS (BCD8sp) technology and use it to benchmark 7 different ML algorithms,

©2025 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.
Published as a conference paper at the NEWCAS2025. DOI:10.1109/NewCAS64648.2025.11107059

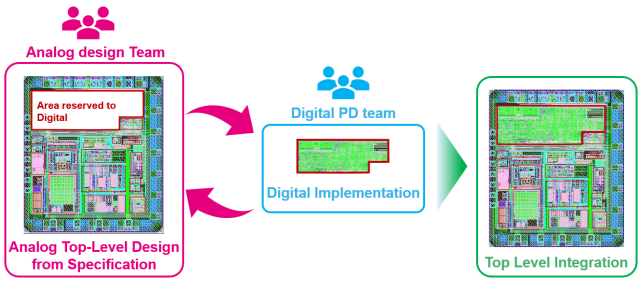


Fig. 1: Analog-on-Top design of an AMS IC. Several iterations between digital and Analog design teams are required so that the top-level integration meets the design specifications.

showing that we can achieve up to 94.8% F1 score with an ensemble of models. Additionally, we explore the impact that the design characteristics have on the prediction quality, showing how the number of routing layers, the row utilization, and the shape regularity of the area mostly impact the model decision.

II. Methods

A. Dataset

The dataset we use in this work consists of 106 PnR runs obtained from 47 digital designs implemented in BCD technology. The considered digital blocks are taken from real-world power management integrated circuits (36%), micro-electromechanical systems (25%), or DCDC converters (25%). On average, the designs feature 42k gates. For each of them, we parse the logs of the EDA tools, extracting the following relevant features, available before executing the flow: **Standard Cells Area**, i.e. the total area occupied by the standard cells as imported from the input Verilog netlist; **Placement Area**, i.e., the die area available for standard cells placement; **#Routing Layers** and **#Ports**, respectively the number of available metal layers and the number of input/output block pins; **Max Frequency**, i.e., the maximum operating frequency of the design; **Max Pin Density**, i.e. the highest pin count per unit length on any side of the block; **#FF**, the number of flip-flops in the design.

We then manually compute two additional features, using as reference the ones proposed in [9]. The first is the **Row Utilization**, i.e. the ratio between the standard cells area and the placement area before the start of the PnR flow. The second is the **Shape Factor**, a similarity score between the shape of the area available and a square. This feature is computed with the following equation:

$$SF = \frac{4A}{P} \tag{1}$$

where A and P denote respectively the area of the die and its perimeter. Both values are obtained after normalizing the sides of the polygon w.r.t. the longest one. Intuitively, the closer this factor is to one, the more the shape resembles a square. Shapes that are closer to a square are preferable for PnR because they minimize routing congestion issues, such as those occurring near elbows and corners, thereby allowing for better utilization of routing resources.

Based on the outcome of the PnR flow, we label each run either as *passing* (true label) or *failing* (false label). A passing label indicates that the digital cells are allocated within the given area without overlaps and without any DRC errors in the final detailed routing phase.

B. ML Algorithms and Metrics

We train and test different ML models for this problem, using a variant of the standard leave-one-out cross-validation. Our dataset contains multiple samples relative to the same design (but different PnR runs with varied placement area, pin positioning, etc). These samples should not be mixed between training and test set to avoid information leakage. So, calling N the number of different designs in the dataset, we train the ML model on $N-1$, and use all samples relative to the remaining design as test set. We then repeat the process rotating the test design. Finally, we compute the model scores by combining the predictions on all test folds. This approach ensures that our results are not split-dependent, as not all designs are equally complex to classify.

For each model, we report the F1 score, i.e. the harmonic mean between precision and recall, and the brier loss, i.e. the mean squared difference between the predicted probability and the actual outcome, as a measure of the model's calibration. Well-calibrated models are desirable so that, in case of low-confidence predictions, an experienced engineer can examine the design, to overrule or confirm their output.

Next, we describe the ML models benchmarked in this work, together with the hyper-parameters explored. For more details on each model, we refer readers to [10].

Decision Tree (DT). We train a Decision Tree using the CART algorithm [11] for comparison with [9]. We explore tree depths ranging from 1 to 10 or with no constraint, as it is the hyper-parameter mostly affecting the prediction quality.

Gaussian Naive-Bayes (GaussianNB). This is the simplest model we explore, i.e., a probabilistic algorithm that applies Bayes' theorem to estimate the samples' probabilities. As it naively assumes conditional independence between every pair of features, its predictive performance is quite limited. Consequently, it is used mainly as a baseline for more complex models in our analysis.

K-Nearest Neighbor (KNN). We use the KNN classifier as a representative of instance-based ML algorithms. We explore values of K ranging from 1 to 10 and the inverse of the Euclidean or Manhattan distances as similarity metrics.

Multilayer Perceptron (MLP). The MLP is the only artificial neural network we benchmark in this work. We explore one and two hidden layers, setting the number of neurons to 25, 50, or 100. We use an LBFGS optimizer, setting the maximum number of calls of the loss function to 15k.

Random Forest (RF). RFs are ensembles of DTs, each trained on random subsets of the data and on a random subset of the features. We explore trees' depths from 1 to 10 or unconstrained and number of trees from 2 to 100.

Support Vector Machine (SVM). An ML model that constructs a set of hyperplanes in a high-dimensional space to separate data into classes. We benchmark linear, radial basis function, sigmoid, and polynomial kernels. For the latter, we explore degrees from 2 to 5.

Ensemble. Lastly, we also consider an ensemble of the models described above to improve classification performance, which simply selects the predicted class using majority voting. After performing hyper-parameters optimization, we apply a feature selection step to our models, to remove features that may lead to overfitting, possibly reducing the prediction quality of unseen data (i.e. different designs). To select the features we use their ANOVA F-value, which measures their relationship with the target variable. Notably, the main advantage of this approach is that it is independent of the ML algorithm employed, allowing a fair comparison among different models.

III. Results

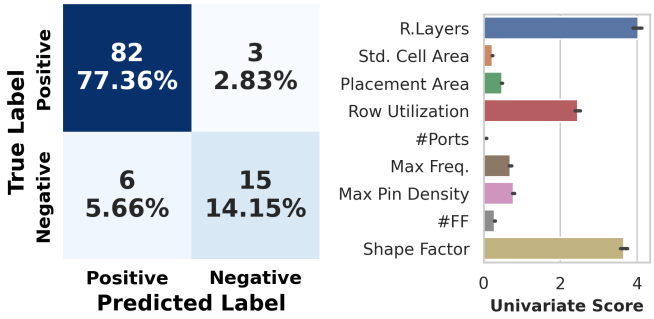
A. Models Exploration

As introduced in Sec. II, we perform a grid search over the main hyper-parameters of each classifier, scoring each configuration with our leave-one-design-out approach. Table I reports the results of the top-scoring models and corresponding hyper-parameters.

TABLE I: Top Models.

Model	F1 [%]	Brier	Hyper-parameters
DT	87.2	0.213	Depth=6
GaussianNB	79.5	0.24	
KNN	90.8	0.149	Neighbours=2
MLP	93.7	0.105	Neurons=100
RF	89.9	0.179	Depth=4 #Trees=2
SVM	91.8	0.125	Reg. Const.=100 Degree=3
Ensemble	94.8	n.a	

A DT with a maximum depth of $D = 6$ obtains an F1 score of 87.2%, while deeper trees overfit the training data. Moreover, a Brier loss of 0.21 denotes that the feasibility probability predicted by the model is often unreliably over-confident. The GaussianNB yields an F1 score of 79.5% and a Brier score of 0.24, thus being the least accurate classifier in our pool. Intuitively, the simple nature of the algorithm fails to capture intricate relations between design features and label. Indeed, an inherent limitation of this classifier lies in its assumption of conditionally independent variables, which is not true for our problem. For instance, the initial row utilization and the std. cell area have a non-linear correlation. The KNN yields a F1 of 90.8% and a loss of 0.149, obtained considering the 2 nearest points in the training data using the Manhattan distance, significantly improving w.r.t. DTs. RFs also improve w.r.t. DTs, limiting their overfitting issues thanks to their bagging and random feature selection mechanisms. A RF of 2 DTs with a maximum depth of 4 achieves 89.9% F1 score and 0.179 Brier loss. SVMs and MLPs achieve the highest F1 scores, 91.8% and 93.7%, respectively, while also attaining the lowest Brier losses (0.125 and 0.105).



(a) Ensemble Confusion Matrix (b) Features Importance

Fig. 2: Results Analysis

Notably, a MLP with a single hidden layer of 100 neurons, is the most accurate individual model, thanks to its ability to approximate non-linear functions. Lastly, an ensamble of all models achieves an F1 score of 94.8%, +1.1% w.r.t. the MLP. Since the Ensemble uses a discrete majority voting mechanism, it does not output a feasibility probability, hence its Brier score cannot be computed.

Figure 2a presents the confusion matrix for the Ensemble of Table I. The highly unbalanced nature of the dataset leads to a model that is more accurate on the positive class, i.e., the one corresponding to designs for which the PnR flow is expected to succeed. Specifically, the Ensemble labels correctly 82 out of 85 designs that will end in a feasible placement. On the other hand, our model is less accurate when predicting the negative class. Out of 21 failed PnR runs, 15 are labeled correctly, while 6 are predicted as passing designs.

In the context of a top-level Analog layout placement optimization, false negative predictions will result in oversizing the area required for the PnR of Digital blocks. Conversely, false positive predictions may lead to under-sizing. Therefore, false positives, although undesirable, simply result in a "wasted" PnR run that will have to be repeated, whereas false negatives (very rare with our best model) are more critical, as they can result in unnecessary increases in silicon manufacturing costs. We leave as future works the study of data augmentation techniques to boost the model performance on the minority class, possibly helping the classifier in reducing false positives.

Interpretability is a key feature of a method like ours, as it allows designers to understand *which design feature(s)* influence the models' prediction (prior to the actual PnR). While some of our more accurate models (e.g. MLP) are black-boxes, our ensembling approach also includes interpretable models such as DTs, which rely on explicit decision rules. Therefore, although less accurate, the latter can be used to provide results' interpretation and feedback to analog designers, e.g. suggesting which features of a design may be the likely cause of a failed PnR prediction (e.g. too high initial utilization, etc).

When compared to the DT proposed in [9], which obtains 79.2% accuracy, our Ensemble outperforms it by 4.7% on the same metric. Moreover, our leave-one-design-out training approach is more robust than the one proposed in [9], avoiding

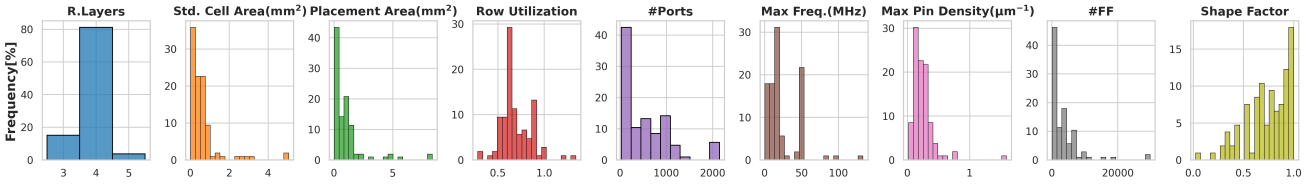


Fig. 3: Features Distributions

scoring on folds with a limited number of designs.

Lastly, we quantify our approach's time advantage w.r.t. a full PnR flow. Our Ensemble requires $< 1s$ for a prediction, regardless of design complexity. Extracting the required input features is also practically instantaneous, as they are all available as reports when launching the EDA tool (before starting PnR execution). In contrast, a full PnR flow can take anywhere from minutes to days, depending on complexity (average $\approx 1h$ on our dataset's designs). Thus, bypassing an infeasible PnR run can save significant time, while the additional inference step incurs negligible overheads when PnR is feasible.

B. Feature Selection

Figure 2b shows the average feature importance across all folds obtained as described in Sec. II. Specifically, we report -$\log_{10}$(p) where p is the p-value, so that features with larger values denote higher importance. As shown, the three most important features are the number of Routing Layers (R.Layers), the Shape Factor, and the Row Utilization, achieving respectively a score of 4.02, 3.6, and 2.45.

Other features are significantly less relevant, with the fourth one being the Max Pin Density at 0.78, three times lower than the Row Utilization. Max Frequency, Std. Cell Area, Placement Area, and #FF yield comparable scores. The least important feature is the #Ports, achieving a 0.04 score, an order of magnitude lower than the Std. Cell Area.

Figure 3 shows the value distributions of the features of Figure 2b. Notably, the distributions of the values of the Shape Factor and the Row Utilization features are among the most uniform in the dataset, making them highly informative for our task. R.Layers instead has only three highly informative levels, with 5 layers having only passing samples and 3 layers leading to failing layouts 56.25% of the time. Other features that have a strong impact on PnR closure such as the design frequency, achieve a lower importance score due to the limited range of values present in the dataset. This shows that more accurate results could be obtained with a richer dataset.

Considering the feature importance shown in Figure 2b, we re-train all classifiers shown in Table I, removing features one by one (starting from the least significant ones). We show the obtained results in Figure 4.

DTs' score ranges from 72.2% to 89%, obtained respectively with the top two and top six features. Notably, increasing the feature count from one to two significantly decreases the F1 score from 84.2% to 72.2%, as relying solely on the shape factor and R.Layers makes the model overconfident in predicting a passing design. However, adding more features reverses this trend, restoring model balance.

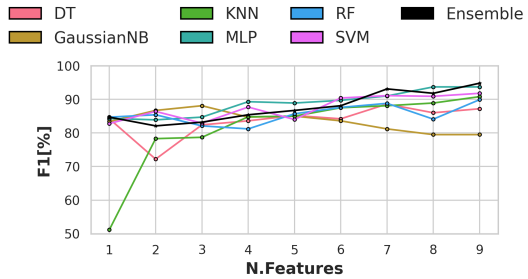


Fig. 4: Feature selection results.

GaussianNB models achieve 88.1% F1 score with the top three features, increasing by 8.6% w.r.t. using all features. Intuitively, the top three features are uncorrelated with each other, the ideal condition for this classifier.

KNNs improve their score when adding features, reaching the top score when using all 9 features. As this classifier computes the distance from similar designs in the training data, it uses any additional feature for more accurate predictions.

MLPs achieve iso-score w.r.t. the starting model when using the top-8 features. Noteworthy, the 8-features model achieves a slight reduction in the Brier loss, going from 0.105 to 0.101, indicating better calibration. Both RF and SVM achieve the maximum F1 score with all 9 features (89.9% and 91.8% respectively), although using only 7 features yields a marginal 0.7% drop for the SVM. Similarly, the Ensemble achieves the top F1 score with 9 features (94.8%), with a slight degradation (-1.7%) with only 7 features. Interestingly, the lowest Brier loss is obtained by the SVM with 6 features, yielding 0.117 loss and 90.4% F1 score, indicating that larger SVMs are more accurate when predicting classes but their prediction confidence becomes less reliable.

IV. Conclusion

In this work, we present an extensive benchmark of various Machine Learning (ML) approaches to automatically estimate the feasibility of placing PnR digital blocks within a given area. On a dataset of real-world designs, we obtain 94.8% F1 with an ensemble of models. Our work presents a reliable and easy-to-implement methodology that accelerates the interaction between Analog and Digital physical designers, reducing the time-to-market for the design phase of AMS ICs.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101007730.

REFERENCES

[1] M. P.-H. Lin, Y.-W. Chang, and C.-M. Hung, "Recent research development and new challenges in analog layout synthesis," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2016, pp. 617–622.

[2] J. Scheible and J. Lienig, "Automation of analog ic layout: Challenges and solutions," in *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, 2015, pp. 33–40.

[3] F. J. Kurdahi and A. C. Parker, "Plest: A program for area estimation of vlsi integrated circuits," in *23rd ACM/IEEE Design Automation Conference*. IEEE, 1986, pp. 467–473.

[4] ——, "Techniques for area estimation of vlsi layouts," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 1, pp. 81–92, 1989.

[5] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee, "Accurate area and delay estimators for fpgas," in *Proceedings 2002 Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 2002, pp. 862–869.

[6] B. Li and P. D. Franzon, "Machine learning in physical design," in *2016 IEEE 25th conference on electrical performance of electronic packaging and systems (EPEPS)*. IEEE, 2016, pp. 147–150.

[7] J. Froemmer, Y. Gowayed, N. Bannow, W. Kunz, C. Grimm, and K. Schneider, "Area estimation framework for digital hardware design using machine learning," in *MBMV 2020-Methods and Description Languages for Modelling and Verification of Circuits and Systems; GMM/ITG/GI-Workshop*. VDE, 2020, pp. 1–10.

[8] E. Zennaro, L. Servadei, K. Devarajegowda, and W. Ecker, "A machine learning approach for area prediction of hardware designs from abstract specifications," in *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 2018, pp. 413–420.

[9] G. Faraone, F. Daghero, E. Serianni, D. Licastro, N. D. Carolo, M. Grosso, G. A. Franchino, and D. J. Pagliari, "Machine learning-based feasibility estimation of digital blocks in bcd technology," 2024. [Online]. Available: https://arxiv.org/abs/2410.07989

[10] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.