MetaCirc: A Meta-learning Approach for Statistical Leakage Estimation Improvement in Digital Circuits

Nouduru Venkata Raghavendra, Deepthi Amuru, Zia Abbas
Centre for VLSI and Embedded Systems Technology (CVEST)
International Institute of Information Technology, Hyderabad (IIIT-H), Hyderabad, India
E-mail: (nouduru.venkata, deepthi.amuru)@research.iiit.ac.in, zia.abbas@iiit.ac.in

2024 IEEE International Symposium on Circuits and Systems (ISCAS) | 979-8-3503-3099-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISCAS58744.2024.10558185

***Abstract*—Aggressive scaling down of transistor dimensions has made process-aware circuit modeling a crucial task. Achieving accurate circuit modeling requires lengthy and resource-intensive simulations. Machine Learning-based surrogate models, offering computational efficiency and speed, are viable alternatives to traditional simulators. This paper introduces a meta-learning approach designed to accurately capture process-induced variations in the leakage power of VLSI circuits. The impact of a wide range of fluctuations in operating conditions, including temperature (-55°C to 125°C) and supply voltage (±10%) has also been incorporated for leakage modeling. The proposed meta-learning model is versatile, enhancing the performance of underlying baseline machine-learning models while eliminating the need for time-consuming hyperparameter optimization. Our experiments on leakage estimation using 16 and 7 nanometer FinFET technology nodes demonstrate an average improvement of up to 50% and 48% in Mean Absolute Percentage Error compared to stand-alone baseline models.**

***Index Terms*—Meta-learning, Machine Learning, Circuit modeling, Statistical Leakage Analysis, Deep Learning**

I. Introduction

Recent advancements in VLSI (Very Large Scale Integration) technology, particularly in the sub-nanometer regime, have created a critical demand for precise circuit behaviour modeling. Shrinking transistor dimensions significantly surge process variations, leading to a substantial increase in leakage power. These manufacturing process variations introduce an element of randomness in the physical structure of devices. When coupled with environmental fluctuations, they exert a notable influence on the electrical characteristics and overall performance of digital circuits [1]. It is imperative to comprehend this behaviour by accurately and efficiently estimating leakage power under various probable processes and environmental conditions before circuit fabrication. Traditionally, precise estimations are attained through Monte-Carlo (MC) simulations employing tools like HSPICE [2]. While these simulations yield accurate results, they are both resource-intensive and time-consuming [3]. To address this issue, efficient surrogate models have been proposed. These models can approximate circuit performance based on statistical parameters, significantly reducing computational demands and substantially shortening the design-to-fabrication time needed for IC (Integrated Circuit) development [4]. Many studies have reported faster computational speed-ups than simulations based on SPICE [4]–[7].

Most approaches focus on efficient simulation approximations through fine-tuning models to a particular dataset. This results in a lack of generalizability, necessitating fine-tuning models after re-training for every new dataset. Additionally, this approach mandates extensive exploration of a vast parameter space to identify optimal hyperparameters. In contrast, our method employs an outer model proficient in enhancing the performance of non-optimized baseline models. The meta-model is trained to deliver precise predictions based on the output of these non-optimized baseline models. To our knowledge, this is the first work utilizing meta-learning for Process, Voltage, and Temperature (PVT) aware leakage estimations.

II. Related Works

Amuru et al. [4] proposed a computationally efficient method to estimate leakage power and propagation delay under PVT variations using gradient boosting. In [6], a multivariate polynomial regression-based approach was proposed to estimate the leakage and total power of digital circuits from statistical process variations. Janakiraman et al. [7] proposed an Artificial Neural Network (ANN) based framework for statistical leakage analysis. Garg et al. [8] demonstrated leakage modeling using Support Vector Machines for 45nm CMOS technology. Sanyal et al. [9] introduced the STABLE algorithm to accurately estimate leakage current using the Newton-Raphson method and incorporating self-loading effects. Apart from modeling circuit leakage power and propagation delay, machine-learning algorithms have been utilized for optimizing digital circuits [10] and at other abstraction levels of VLSI design [11]. Machine-learning models, on their own, don't generalize well. Meta-learning techniques are commonly used to distil knowledge from multiple learning instances, thereby augmenting the capacity to generalize across a range of tasks [12]. These algorithms have found extensive application in enhancing performance across diverse tasks, including weight initialization (MAML) [13], neural architecture search [14], domain generalization [15], and hyper-parameter optimization for few-shot learning (MetaSGD) [16], among others.

III. Methodology

A. Proposed hypothesis

Meta-learning has proven effective in enabling generalization across diverse tasks. The presented approach utilizes neural networks to implement a meta-learning strategy for making subtle decisions between pre-trained models. The

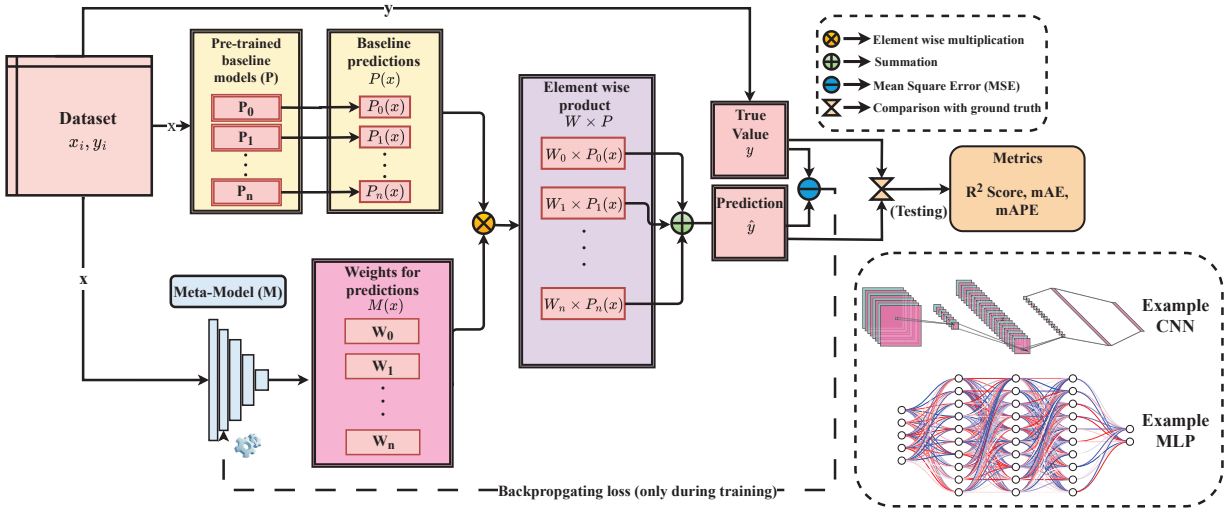


Fig. 1: Block level overview of MetaCirc algorithm

underlying assumption is that the pre-trained models exhibit satisfactory performance (R^2 Score > 0.9) and are arranged in descending order based on their R^2 Score. Experimental results demonstrate that the suboptimal performance of pre-trained models hinders the convergence of the meta-model.

The implementation encompasses both a meta-model and a collection of baseline pre-trained models tailored for a specific dataset. The meta-model's role is to improve the predictions made by the pre-trained models without the need to optimize their hyperparameters. A high-level schematic of the proposed algorithm is depicted in Fig 1. The pre-trained models are trained to predict leakage power for a specific dataset. Meanwhile, the meta-model, which is a deep learning network, learns to allocate weights to the predictions generated by these models for a given data point. This approach enables the meta-model to discern which model performs optimally for a particular PVT value.

$$W_i = M(x) \tag{1}$$

$$meta\_pred(\hat{y}) = \sum_{i=0}^{n} W_i \times P_i(x) \tag{2}$$

Initially, the pre-trained models $(P_0, P_2, ..., P_n)$ undergo training on the dataset using default parameters. Subsequently, the meta-model (M) is trained to assign a weight to the prediction produced by each pre-trained model given a specific input. The model learns to dynamically generate a total of "n" weights $(W_1, W_2, ..., W_n)$ for a given PVT input x, each weight W_i being associated with a pre-trained model P_i. The output $(\hat{y})$ is a weighted summation defined in Equ. (2), where x represents the PVT input provided to both the meta-model and the set of pre-trained models. Here, $P_i(x)$ signifies the prediction generated by the i^{th} pre-trained model. Similarly, the weights are determined by the meta-model as outlined in Equ. (1). It's important to note that these weights do not pertain to the internal weights of the network, but rather, they represent the outputs of the final layer of the meta-model (M).

B. Implementation

1) Data pre-processing: Given the significantly small magnitude of leakage values for FinFETs (approx 10^{-8} scale), it was imperative to rescale the ground truth values to facilitate model convergence to an optimal state. The model input, representing PVT variations, underwent standard scaling to transform it into a normally distributed form with a mean of zero and a standard deviation of one. As for the ground truth values (leakage power), a combination of log scaling followed by standard scaling was employed to adjust the skewness in its distribution. Subsequently, the predictions were reverted to their original scale through an inverse scaling process. Such standardization technique has been demonstrated to enhance model performance [17].

Algorithm 1 describes the pre-training and training of baseline models and meta-models. Actual leakage values for various PVT variations were obtained by conducting HSPICE simulations for each circuit [4]. The baseline models (pre-trained models) were trained on each dataset without undergoing hyperparameter tuning. Subsequently, these pre-trained models were arranged in descending order based on their R^2 on the training set, separately for each dataset. For each dataset, meta-models were initialized, and their training process is outlined in Algorithm 2. The mean square error (MSE) serves as the loss criterion during the backpropagation of the meta-models. A train-test split was executed with a ratio of 95% for training and 5% for testing. All the data-preprocessing steps described above were carried out using the Pandas [18], [19], Numpy [20], and Scikit-Learn libraries [21]. The baseline models were implemented using Scikit-Learn library [21], and deep learning models (meta-models) using the PyTorch framework [22].

2) Predictions and model testing: Algorithm 3 provides a comprehensive overview of the MetaCirc approach. Initially, the testing data, the corresponding pre-trained models $(P_0, ..., P_n)$, and the trained meta-model "M" are retrieved. The predictions of the pre-trained models for the current

data point are calculated and stored in "pre-preds" ($P(x)$). Subsequently, the data point is inputted into the meta-model to generate weights for each of the pre-trained models' predictions. These weights are then applied to the predictions, and their sum yields the final prediction, as illustrated in Equ. (1).

Algorithm 1 Model Training

```
Perform HSPICE simulations to create datasets
Initialize baseline models
for all Datasets do
    Fetch training data
    for i = 0 to n do
        Train baseline model P_i
        Report metrics (R2 Score, mAPE)
    end for
    for i = 0 to k do
        Train meta-model M_k
        Report metrics (R2 Score, mAPE)
    end for
end for
```

Algorithm 2 Meta model prediction & backpropagation

```
Fetch pre-trained models & dataset
Initialize Meta-model "M"
for i = 0 to num_data_points do
    x, y = dataset[i] {x, y → PVT, ground truth}
    P(x) = P_0(x), P_1(x), ..., P_n(x) {P(x) → Pre-trained predictions}
    W_0, W_1, ..., W_n = M(x) {Weights for P(x)}
    ŷ = W_0 × P_0(x) + W_1 × P_1(x), ..., W_n × P_n(x) {Eq. 1}
    loss = MSE(ŷ, y)
    Backpropagate loss
end for
```

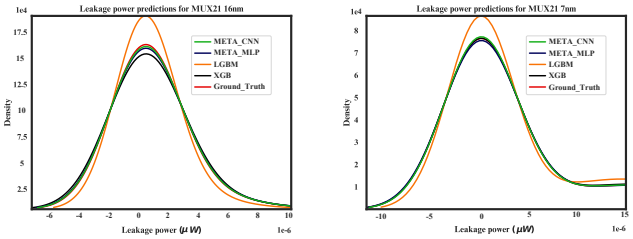
Algorithm 3 Overall algorithm

```
for all datasets do
    Fetch pre-processed testing data, corresponding pre-trained models (P), and meta-model (M)
    for i = 0 to num_data_points do
        x = testing_data[i]
        ∀ i ∈ [0, n] pre_preds = P_i(x)
        W_0, ..., W_n = M(x) {W → Weights}
        Prediction (ŷ) = Σ_{i=0}^{n} W_i × P_i(x)
        ŷ = exp(rescale(ŷ))
    end for
end for
```

IV. Results & Analysis

A. Experimental setup

HSPICE was utilized to conduct simulations for standard cells in both the 16nm and 7nm FinFET technology nodes



(a) MUX21 16nm (b) MUX21 7nm

Fig. 2: Comparison between predictions of baseline and meta-models

[23]. The simulation setup employed to generate the dataset aligned with the one detailed in [4]. In total, 13 standard cells were used to test the algorithm across both technology nodes, resulting in 26 datasets. Two pre-trained models and two meta-models were employed for each dataset to conduct a thorough analysis. A total of 104 models were trained and evaluated. The pre-trained models selected were Light Gradient Boosting Model (LGBM) [24] and XGBoost model (XGB) [25]. These choices were motivated by their efficiency in covering the dataset using random weighted sampling without replacement [26] while training weak learners. This characteristic significantly enhanced the convergence rate of the meta-model. As for the meta-models, a simple Multi-Layer Perceptron (MLP) with 3 hidden layers (each of size 2000) and Convolutional Neural Network (CNN) with 3 convolutional layers (kernel size = 3 and stride = 1) and 2 linear layers (each of size 500) were chosen. LeakyReLU activation was used along with Sigmoid activation at the last layer. These deep learning models are adept at fitting complex functions, rendering them suitable as meta-learners. Due to the shared feature representation of CNNs, they train faster than MLPs in many cases. LGBM and XGB were trained with default parameters. The CNN comprised three convolutional and two linear layers, while the MLP featured three hidden layers. Evaluation metrics included R^2 Score [27] (Equ. 3), mean Absolute Percentage Error (mAPE) (Equ. 4), and mean Absolute Error (mAE) (Equ. 5). Each meta-model underwent training for 30 epochs with a batch size of 256. The total training time was approximately 2-3 hours.

$$R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y})^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2} \tag{3}$$

$$mAPE(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i-\hat{y}_i|}{\max(\epsilon,|y_i|)} \tag{4}$$

$$mAE(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|\,. \tag{5}$$

B. Results and Discussion

Tables I, III display the performance of pre-trained (baseline) models and II, IV show meta-models' performance on

the leakage datasets of 16nm and 7nm FinFET technologies respectively. At 16nm node (Tables I, II), 2x1 multiplexer cell showed a significant improvement where its mAPE reduced from 17.44% to 4.61%, resulting in 73.5% improvement in leakage estimation (% improvement is calculated using Equ. 6). Likewise, the best reduction in mAPE at the 7nm node (Tables III, IV) was observed for the NAND2 gate, from 17.02% to 4.63% (around 73% improvement). The best improvement is highlighted in these tables. The proposed MetaCirc approach shows improvement in all the metrics - R^2, $mAPE$, and mAE across all the datasets in both the technology nodes, demonstrating its efficiency. On average, mAPE is reduced by 4% in all the datasets, resulting in around 50% better leakage estimation than the baseline models. Figures 2a and 2b illustrate a plot comparing the predictions of pre-trained and meta-models against the ground truth for the 16nm and 7nm 2x1 multiplexer datasets, respectively. These graphs demonstrate that the meta-models effectively align with the ground truth distribution compared to the baseline models.

$$\%\ improvement = \frac{mAPE_{base} - mAPE_{meta}}{mAPE_{base}} \times 100 \quad (6)$$

TABLE I: Pre-trained model performance for 16nm node

Standard Cell	Model Name and Metrics					
	LGBM			XGB		
	R2	mAE	mAPE (%)	R2	mAE	mAPE (%)
AND2	0.9660	9.8E-10	7.6581	0.9613	1.2E-09	10.0106
NAND2	0.9570	7.2E-10	10.5886	0.9510	7.3E-10	10.5633
NOR2	0.9863	2.2E-10	5.7239	0.9899	2.2E-10	5.7771
OR2	0.9904	4.1E-10	4.0500	0.9902	4.6E-10	4.5749
XNOR2	0.9904	1.5E-09	4.2091	0.9891	1.7E-09	4.7609
XOR2	0.9820	1.8E-09	5.3808	0.9866	1.8E-09	5.5119
AND3	0.9521	1.6E-09	8.2606	0.9681	1.6E-09	8.6951
AO12	0.9854	1.0E-09	4.3039	0.9909	1.1E-09	4.5665
FA	0.9776	3.4E-09	4.4699	0.9865	3.5E-09	4.7518
MUX21	0.8775	1.5E-07	**17.4416**	0.9076	1.4E-07	15.1702
NAND3	0.9486	8.2E-10	8.7507	0.9460	9.8E-10	10.8688
NOR3	0.9598	3.4E-10	8.5315	0.9479	2.9E-10	6.5984
OR3	0.9672	6.0E-10	4.9510	0.9815	6.6E-10	5.6982

TABLE II: Meta-model performance for 16nm node

Standard Cell	Model Name and Metrics					
	META CNN			META MLP		
	R2	mAE	mAPE (%)	R2	mAE	mAPE (%)
AND2	0.9949	4.6E-10	4.3341	0.9957	4.2E-10	3.9282
NAND2	0.9936	2.8E-10	4.9875	0.9957	2.3E-10	4.3823
NOR2	0.9956	1.5E-10	3.6429	0.9963	1.5E-10	3.4591
OR2	0.9944	3.2E-10	3.0125	0.9955	2.9E-10	2.8160
XNOR2	0.9938	1.3E-09	3.4939	0.9951	1.2E-09	3.3391
XOR2	0.9936	1.2E-09	3.6992	0.9942	1.1E-09	3.5571
AND3	0.9953	6.1E-10	3.4377	0.9955	5.7E-10	3.2287
AO12	0.9984	4.7E-10	1.9618	0.9985	4.5E-10	1.8637
FA	0.9973	1.7E-09	2.4567	0.9975	1.4E-09	2.1345
MUX21	0.9958	2.9E-08	4.8978	0.9948	3.2E-08	**4.6161**
NAND3	0.9964	2.7E-10	3.5893	0.9967	2.7E-10	3.4229
NOR3	0.9899	1.6E-10	3.4891	0.9907	1.4E-10	3.1270
OR3	0.9863	3.6E-10	2.5617	0.9943	3.0E-10	2.2991

The analysis above indicates that the MetaCirc algorithm can bring predictions closer to the actual values. While it is possible to fine-tune the baseline models to enhance their performance, optimizing their hyperparameters demands substantial resources. In contrast, training meta-models on top of the baseline models requires fewer resources and effort. This approach offers a dual advantage as the meta-models don't directly predict values but instead learn to select the appropriate pre-trained models based on the input. This enables the meta-models to generalize effectively to variations and improve predictions on data they were not explicitly trained on. In contrast, when hyper-parameter optimization is applied to a model on a dataset, the parameter search space expands exponentially with the number of features. Employing a straightforward brute-force method significantly prolongs the time required for fine-tuning models.

TABLE III: Pre-trained model performance for 7nm node

Standard Cell	Model Name and Metrics					
	LGBM			XGB		
	R2	mAE	mAPE (%)	R2	mAE	mAPE (%)
AND2	0.9690	5.5E-10	7.4642	0.9635	6.2E-10	8.6824
NAND2	0.9625	4.4E-10	11.3121	0.9037	6.7E-10	**17.0232**
NOR2	0.9921	8.5E-11	3.3509	0.9925	9.2E-11	3.3160
OR2	0.9865	2.4E-10	3.9311	0.9871	2.7E-10	4.6385
XNOR2	0.9824	9.7E-10	4.2486	0.9887	1.0E-09	4.8047
XOR2	0.9796	8.9E-10	4.6036	0.9798	9.0E-10	4.5415
AND3	0.9547	9.4E-10	7.8928	0.9627	9.3E-10	8.1777
AO12	0.9791	6.5E-10	4.3837	0.9918	6.2E-10	4.6124
FA	0.9810	2.0E-09	4.7983	0.9853	2.0E-09	4.9518
MUX21	0.9356	9.0E-07	13.7656	0.9815	4.9E-07	8.6721
NAND3	0.9419	6.0E-10	9.8906	0.9597	5.3E-10	9.2580
NOR3	0.9652	1.3E-10	4.1652	0.9825	1.3E-10	3.8679
OR3	0.9737	3.4E-10	4.6873	0.9773	3.8E-10	5.5700

TABLE IV: Meta-model model performance for 7nm node

Standard Cell	Model Name and Metrics					
	META CNN			META MLP		
	R2	mAE	mAPE (%)	R2	mAE	mAPE (%)
AND2	0.9934	2.9E-10	4.5437	0.9953	2.5E-10	4.0210
NAND2	0.9958	1.5E-10	4.9010	0.9960	1.5E-10	**4.6393**
NOR2	0.9938	8.0E-11	2.2958	0.9937	8.0E-11	2.1151
OR2	0.9947	1.7E-10	2.7188	0.9948	1.7E-10	2.6202
XNOR2	0.9946	7.1E-10	3.4204	0.9945	6.8E-10	3.2476
XOR2	0.9928	6.0E-10	3.4036	0.9921	6.1E-10	3.3666
AND3	0.9952	3.5E-10	3.3690	0.9956	3.4E-10	3.2652
AO12	0.9979	3.1E-10	2.3123	0.9971	3.1E-10	2.2089
FA	0.9968	9.7E-10	2.5367	0.9976	8.4E-10	2.1837
MUX21	0.9978	1.6E-07	3.8445	0.9975	1.7E-07	3.7396
NAND3	0.9960	1.9E-10	3.8378	0.9952	1.8E-10	3.6328
NOR3	0.9900	9.2E-11	2.0271	0.9865	9.3E-11	1.8805
OR3	0.9902	1.9E-10	2.2212	0.9909	1.8E-10	2.1738

V. Conclusion & Future Scope

We introduce an effective meta-learning strategy capable of enhancing the performance of baseline models without the need for hyperparameter optimization. Experimental results conducted on both 16nm and 7nm technology nodes reveal an average improvement of 50% and 48.68% in terms of mAPE, respectively. The MetaCirc algorithm proposed in this study can serve as a versatile tool for refining predictions from pre-trained models without the necessity of fine-tuning hyperparameters of the baseline models. This approach can significantly reduce the complexity of simulations, as meta-models exhibit the capability for few-shot learning, enabling them to adapt to new data with a minimal number of samples. In future research, we aim to extend this algorithm to develop a comprehensive framework of meta-models that can leverage prior experience to estimate leakage power for previously unseen circuits.

Acknowledgment

This work was generously supported by IHUB Data at IIIT Hyderabad.

REFERENCES

[1] A. Zjajo, *Stochastic process variation in deep-submicron CMOS*. Springer, 2016.

[2] M. Olivieri and A. Mastrandrea, "Logic drivers: A propagation delay modeling paradigm for statistical simulation of standard cell designs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 6, pp. 1429–1440, 2013.

[3] D.-W. Kim and T.-Y. Choi, "Delay time estimation model for large digital cmos circuits," *VLSI Design*, vol. 11, no. 2, pp. 161–173, 2000.

[4] D. Amuru, M. S. Ahmed, and Z. Abbas, "An efficient gradient boosting approach for pvt aware estimation of leakage power and propagation delay in cmos/finfet digital cells," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[5] D. Helms, R. Eilers, M. Metzdorf, and W. Nebel, "Leakage models for high-level power estimation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1627–1639, 2018.

[6] D. Amuru, A. Zahra, and Z. Abbas, "Statistical variation aware leakage and total power estimation of 16 nm vlsi digital circuits based on regression models," in *VLSI Design and Test*, A. Sengupta, S. Dasgupta, V. Singh, R. Sharma, and S. Kumar Vishvakarma, Eds. Singapore: Springer Singapore, 2019, pp. 565–578.

[7] V. Janakiraman, A. Bharadwaj, and V. Visvanathan, "Voltage and temperature aware statistical leakage analysis framework using artificial neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 7, pp. 1056–1069, 2010.

[8] L. Garg and V. Sahula, "Variability aware support vector machine based macromodels for statistical estimation of subthreshold leakage power," in *2012 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2012, pp. 253–256.

[9] A. Sanyal, A. Rastogi, W. Chen, and S. Kundu, "An efficient technique for leakage current estimation in nanoscaled cmos circuits incorporating self-loading effects," *IEEE Transactions on computers*, vol. 59, no. 7, pp. 922–932, 2010.

[10] K. Agarwal, A. Jain, D. Amuru, and Z. Abbas, "Fast and efficient resnn and genetic optimization for pvt aware performance enhancement in digital circuits," in *2022 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2022, pp. 1–4.

[11] D. Amuru, A. Zahra, H. V. Vudumula, P. K. Cherupally, S. R. Gurram, A. Ahmad, and Z. Abbas, "Ai/ml algorithms and applications in vlsi design and technology," *Integration*, vol. 93, p. 102048, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926023000901

[12] T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, "Meta-learning in neural networks: A survey," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 44, no. 09, pp. 5149–5169, sep 2022.

[13] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 1126–1135. [Online]. Available: https://proceedings.mlr.press/v70/finn17a.html

[14] W. Li, S. Gong, and X. Zhu, "Neural operator search," *Pattern Recognition*, vol. 136, p. 109215, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132032200694X

[15] Y. Chen, P. Song, H. Liu, L. Dai, X. Zhang, R. Ding, and S. Li, "Achieving domain generalization for underwater object detection by domain mixup and contrastive learning," *Neurocomput.*, vol. 528, no. C, p. 20–34, apr 2023. [Online]. Available: https://doi.org/10.1016/j.neucom.2023.01.053

[16] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv preprint arXiv:1707.09835*, 2017.

[17] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[18] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3509134

[19] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.

[20] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2

[21] 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.

[22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[23] [Online]. Available: http://www-device.eecs.berkeley.edu/bsim/

[24] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[25] 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, pp. 785–794.

[26] A. J. Ferreira and M. A. T. Figueiredo, *Boosting Algorithms: A Review of Methods, Theory, and Applications*. New York, NY: Springer New York, 2012, pp. 35–85.

[27] S. Wright, "Correlation and causation," 1921.