AUTOMATIC DESIGN OPTIMIZATION OF MICROELECTRONIC POWER SWITCHES

G. Nicolae[1,2], H. Cucu[1], C. Burileanu[1], A. Buzo[2], C. Feuerbaum[2], G. Pelz[2]

Design optimization plays a crucial role in advanced discrete device design as the performance requirements are constantly increasing. For power switches, there is the need of high accuracy of on-state resistance (R_{on}) simulation. To achieve a high accuracy, there is the requirement of a complex 3D finite element simulation, which usually takes tens of minutes. Furthermore, preparing the simulation (e.g. model setup, adjusting design parameters, solving numerical issues, etc.) increases the time for obtaining results additionally. Therefore, designing new devices involves a lot of iterative time-consuming manual work. In this paper, we introduce an automatic design optimization methodology for microelectronic power switches that minimizes the amount of manual work of engineers and the design time of new devices. The methodology is structured in 3 blocks: Automated Simulation Setup, Prediction Metamodel and Design Optimizer. A high accuracy metamodel (error less than 2%) is obtained using only a couple of hundreds of 3D finite element simulation for training. Then, optimal design parameters of any device using the same package technology, which needs to meet an expected value of R_{on}, will be automatically determined in a matter of seconds.

Keywords: device optimization, design optimization, MOSFET, on-state resistance, machine learning

1. Introduction and Related Work

Power electronics semiconductors have attracted much more attention since renewable energies and electric vehicles became high-importance topics in academia and industry. Required performance specifications of the power semiconductor devices are constantly increasing, hence the design process of the new devices is becoming more challenging. Considering that the simulation flow of a package technology is already available, engineers begin an iterative design process for multiple devices with distinct performance requirements. As designers need to find the minimum chip area of the devices to meet the requirements, the number of R_{on} simulations becomes very high. The 3D finite

[1]University Politehnica of Bucharest, Romania, e-mail: georgian.nicolae@ieee.org, horiacucu@gmail.com, corneliu.burileanu@upb.ro

[2]Infineon Technologies, Neubiberg, Germany, e-mail: Andi.Buzo@infineon.com, Christian.Feuerbaum@infineon.com, Georg.Pelz@infineon.com

element simulation time is also high (tens of minutes) because R_{on} must be accurately determined. The time for obtaining simulation results is furthermore increased by the necessary steps for preparing the simulation (e.g. model setup, adjusting design parameters, solving numerical issues, etc.). Therefore, design optimization is time consuming and requires a lot of effort. Although automatic optimization has been employed in various semiconductor design-related tasks, to the best of our knowledge, automatic design optimization of chip area for meeting the R_{on} specification was not investigated in the literature.

Paper [1] presents an optimization framework for GaN Power device design which minimizes the energy loss of a boost converter. In order to enable the optimization the authors have chosen the downhill simplex algorithm for two main reasons: (i) it does not involve determining Hessian matrices, derivatives, or gradients, and (ii) it requires much fewer computations of the cost function. Because evaluating the cost function involves running a simulation, choosing a complex search algorithms would make the entire flow unfeasible because of the time necessary for the simulation to complete.

In [2] is described a neural network approach for modelling and optimizing Si-MOSFET manufacturing. The aim is to adapt the design parameters of the transistor in order to meet the desired figure of merit. The algorithm used as optimizer is Gradient Descend and Si-MOSFET is modeled using Neural Networks at technology level (TCAD).

The authors of [3] employ a multi-objective genetic algorithm for optimization while the cost function is evaluated using a behavioral circuit model of a transistor. Therefore, within thousands of iterations, the genetic algorithm optimizer is able to determine the optimal parameters of the behavioral circuit model to meet the circuit requirements.

In this paper, we propose a design optimization framework for microelectronic power switches. The first step is to develop an automatic simulation framework for a specific package. Starting from a set of design parameters, a machine learning metamodel is trained using simulation data. Then, this metamodel is used in optimization algorithms for determining the optimal design parameter values which meet the device required performance. Technical challenges and proposed solutions are detailed while presenting the optimization flow. Using only a couple of hundreds of 3D finite element simulations to train a high accuracy metamodel (error less than 2%), optimal design parameters of any device using the same package technology, which needs to meet an expected value of R_{on}, will be automatically determined in a matter of seconds.

The paper is structured as it follows. Section 2 illustrates an overview of the optimization flow, while Sections 3–5 detail each functional block. Section 6 presents the results of the optimization applied to two technologies and Section 7 draws concluding remarks.

2. Design Optimization Methodology

Figure 1 illustrates the three main blocks of the proposed methodology. Although each block is detailed in Sections 3–5, a short overview is presented as it follows.

- *Automated Simulation Setup* is the first step towards automatic optimization in the presented methodology. The main function of this block is to evaluate the performance of a device design using traditional 3D finite element simulation. Therefore, the input of this block consists in a design parameter configuration of a device and as output R_{on}.
- *Prediction Metamodel* aims to substitute the Automated Simulation Setup for complex tasks which require a very large number of simulations. In the current case, determining R_{on} using metamodel is million times faster compared to finite element simulation.
- *Design Optimizer* has the role of determining the optimal design parameters which feature the minimum chip area for the required R_{on}.

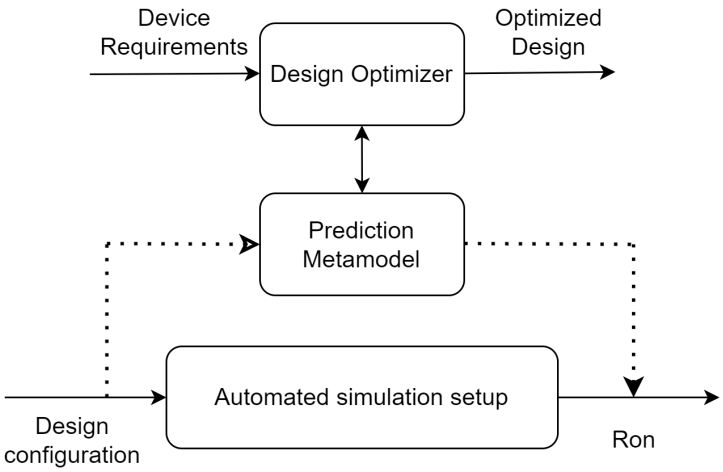

FIGURE 1. Design optimization Overview

3. Automated Simulation Setup

Developing the Automated Simulation Setup is fundamental for achieving optimization. On this level, the design parameters of a power MOSFET design are carefully defined. Therefore, this design will have a set of fixed parameters such as package type, technology parameters, etc., and another set of variable parameters which, in the presented case, is detailed in TABLE 1.

An overview of Automated Simulation Setup is illustrated in Figure 2. The first block has as output a 2D layout based on user input (e.g. chip dimensions, technology, chip position on lead frame, etc.). The second block has the role of translating the 2D layout into a 3D structure, using a set of fixed and variable parameters. A 3D section of the device is illustrated in

Figure 3. The last block is the 3D finite element simulator which takes the 3D layout and determines R_{on}.

The main challenge of this block consists in integrating multiple tools (e.g. collecting user input, preparing chip design, simulation, etc.) in order to ensure consistency of the simulation results. This is achieved by developing a parametric device design and then imposing design rule constraints.

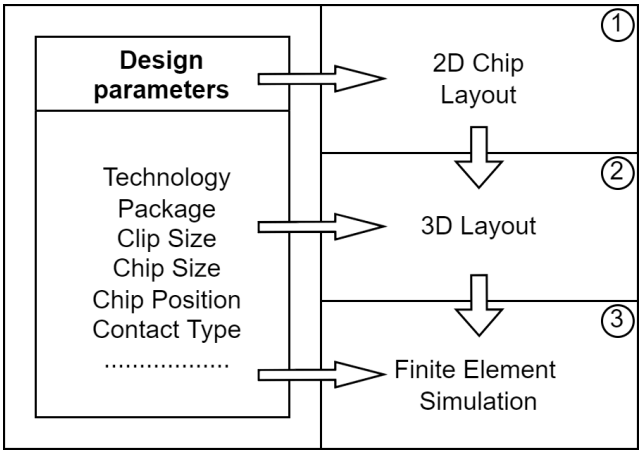

FIGURE 2. Automated Simulation Setup

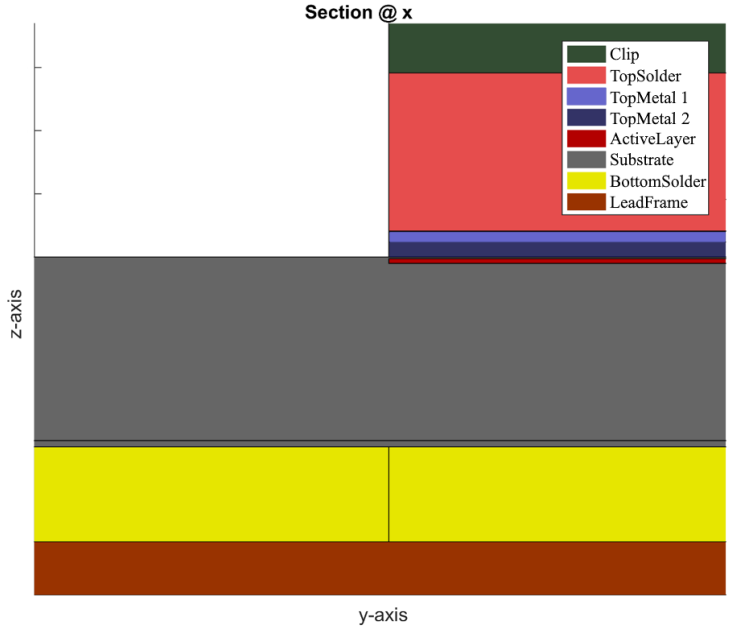

FIGURE 3. Device section on x-axis [4]

TABLE 1. Design parameters [5]

No.	Parameter name	No.	Parameter name
1	Clip Width	6	Clip Thickness
2	Clip Length	7	Top Metal 1 Thickness
3	Chip Width	8	Top Metal 2 Thickness
4	Chip Length	9	Top Solder Thickness
5	Chip Thickness	10	Bottom Solder Thickness

4. Prediction Metamodel

The metamodel enables an R_{on} estimation million times faster than classical 3D finite element simulation, at the cost of an insignificant loss in accuracy. Then, when executing a large number of simulations, the overall simulation times duration becomes trivial. To do this, machine learning techniques are employed. Therefore, the first requirement in training a prediction metamodel is building a simulation dataset.

The trade-off between overall simulation time of the dataset and the coverage of design parameter space needs to be carefully defined, as 3D finite element simulation is time-consuming (e.g. a single simulation takes at least 20 minutes, on an average computer, for a generic power MOSFET design). Reducing the number of simulations in the dataset will result in an accuracy loss of the machine learning metamodel.

In this work, the method used to generate the dataset has two components. The first component aims to identify the corners of the multi-dimensional design parameter space and, therefore, defining the range of values of R_{on}. To ensure this type of sampling, we employ a full factorial design of experiments, illustrated in Figure 4 (a). The second component consists in uniform sampling, illustrated in Figure 4 (b), and has the purpose of covering the design parameter space.

After R_{on} simulation is complete for each entry of the dataset, the next step is training and evaluation of the metamodel. To ensure a reliable evaluation, the simulation dataset is split in two main categories: training data and evaluation data. Determining the best machine learning algorithm and its configuration parameters for modelling is also very important and will be approached in the Results and Use Cases section.

Another challenge in metamodel training is regarding discrete changes in transistor design. As an example, considering a specific technology, the number of gate fingers in the transistor design depends on the dimensions of the chip. This kind of discrete changes should be addressed accordingly, as they represent discontinuities in the performance curve (R_{on}) of the parametric design. This aspect will be covered for *Technology B* in the Results and Use Cases section.

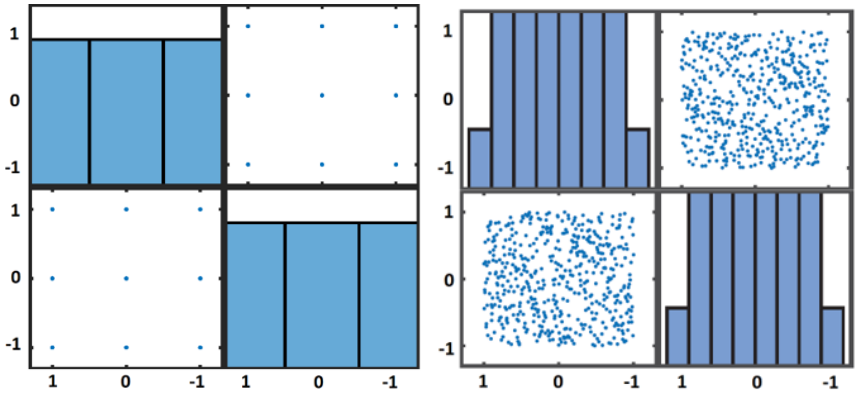

FIGURE 4. Sampling Techniques

(1) *Linear Regression*

Linear regression is the least complex algorithm used for modelling a result variable as a linear function of multiple input variables. The main advantages of this algorithm is simplicity and interpretability. Also, by using this algorithm as a starting point in modelling, we can empirically estimate which machine leaning algorithm would yield high accuracy and also the complexity of such a model [6].

(2) *Support Vector Machine Regression*

Support Vector Machine (SVM) was initially designed for classification tasks as it aims to find the best separation hyperplane between multiple classes. SVM regression aims to find the hyperplane which intersects most of the data points in the training set. Once this hyperplane is determined, the response of any point in the space can be estimated. An advantage of this method consists in the possibility of using different kernels to efficiently map the input variable space [7].

(3) *Gaussian Process Regression*

The Gaussian process is a probabilistic modelling algorithm. The main idea of this method is to fit a distribution over a function. An advantage of this algorithm is the choice of the appropriate covariance functions for the specific target model. In this work have been used Rational Quadratic and Matern 5/2 covariance functions [8].

(4) *Neural Networks*

Neural networks are currently one of the most popular machine learning methods. The main advantage of neural networks is the possibility of fine adjusting the model size for achieving the optimal trade-off between complexity, data volume and accuracy [9]. In this work we evaluated fully connected neural networks consisting in a single hidden layer.

5. Design Optimizer

The purpose of this block is to determine the best values of the design parameters which result in the best R_{on}. This optimization problem consists in minimizing a cost function which has two components: R_{on} deviation and chip area. There are two important aspects to consider. The first is enforcing constraints regarding technology limitations and R_{on} requirement. The second important aspect is to evaluate possible discontinuities in the R_{on} curve as a function of design parameters. From this point of view, if the metamodel response is continuous, the optimization can be performed by a deterministic algorithm. Otherwise, the optimization algorithm must provide robustness with respect to discontinuities in the cost function. In this work, two optimization algorithms have been used: interior-point and differential evolution.

The interior-point algorithm is one of the most popular methods for solving non-linearly constrained optimization problems. This method consists in finding the solution in an iterative way, starting from an initial point. Each iteration involves one of the two main types of steps: Newton step, if projected Hessian is positive definite, or conjugate gradient step, which minimizes a quadratic approximation in a trust region [10].

Differential evolution is an iterative population-based optimizer which starts from a set of randomly selected values. Then, it continues by updating this population based on the scaled difference of two randomly selected population vectors (perturbation). This algorithm has proven to be highly effective when approaching a global optimization problem defined with continuous and/or discrete parameters [11].

6. Results and Use Cases

The optimization methodology is applied to two distinct cases of package technologies. *Technology A* features a continuous parameterization characteristic, the performance parameter R_{on} is a continuous function. *Technology B* is characterized by a discontinuous R_{on} function of design parameters, as the number of gate fingers depends on the chip size. Those discontinuity points raise the challenge of optimization. Details regarding application of the optimization methodology for these two technologies are provided in the following subsections.

6.1. *Technology A* Devices

Technology A is a generic power MOSFET. This inherits all the characteristics described in Section 3. As a specific feature, this technology has the property of being parametrized in a continuous matter. In other words, adjusting the size of each design parameter involves only resizing operations starting from a design template.

The first step is developing the Automated Simulation Setup as presented in Section 4. The main challenge is to ensure data consistency while resizing

the layers and other geometry elements of the transistor design. Then, the simulation dataset needs to cover sufficient data for training the metamodel. Considering that enough data is provided to train a good metamodel, the optimization algorithm will have to be designed or adjusted to handle R_{on} as a function of the design parameter space (Table 1). For covering the parameter space, a number of 1300 of simulations were performed. This data is split between 70% training, 15% validation and 15% evaluation for neural network training and for the other machine learning regressors was used a 10 k-fold validation [5].

The results of the metamodel training are presented in Table 2. The best results are achieved by neural networks, but the Gaussian process also shows promising performance. Taking this into account, the neural network-based metamodel is used further for optimization.

Figure 5 illustrates an optimization example in which the chip area is minimized while achieving a target R_{on}. Three starting points are randomly selected as input for the optimization algorithm and, as it can be observed in the figure, they all lead to the same optimal point, represented with green. The yellow starting point was selected to illustrate that even though the initial set of design parameters does not satisfy technology constraints, the optimizer is able to find the optimal point. The experiments presented were performed using the interior-point optimization algorithm.

6.2. *Technology B* Devices

For *Technology B*, the variation of the design parameters involves two processes: (i) geometry scaling, (ii) adjusting the number of gate fingers. Hence, the performance result R_{on} can be considered as a function which is discontinuous in the points where the second process is involved. Also, being a more complex design, the simulation time is longer compared to *Technology A*. Considering these aspects, sampling design space parameters needs to capture the limits of continuity regions of the R_{on} function. Then, in order to achieve an accurate metamodel, our proposal is to train separate machine learning models for each continuous region of R_{on}.

TABLE 2. *Technology A* Metamodel Evaluation

Algorithm	Maximum relative error [%]
Neural Networks	0.14
GPR Matern 5/2	0.70
GPR Rational Quadratic	1.13
SVM Cubic Regression	5.58
Linear Regression	74.02

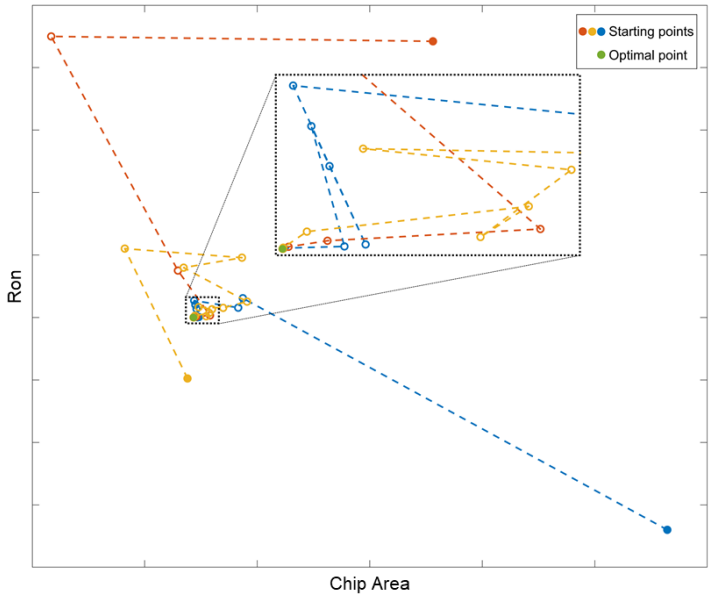

FIGURE 5. Chip Area optimization for *Technology A*

The experiments for *Technology B* feature a reduced set of design parameters to illustrate the relevant application differences at a minimum simulation time cost. The simulation dataset is composed of 500 samples where the parameters 1-5 from Table 1 are varied while the rest are kept constant. In the presented case, the R_{on} response space is divided in 2 continuous regions. Therefore, 100 samples cover a reduced region and the rest of 400 cover the other region. The results of metamodel training are presented in Table 3. Two types of approaches have been tried: (i) training a single metamodel (referred as *Single Model*) for all design parameter space even though it has a discontinuity hyperplane and (ii) training separate metamodels for each continuous regions, referred as *Model 1* for the small subspace and *Model 2* for the largest subspace. For neural networks, the percentage of data used for training, validation and test is the same as used for *Technology A* and 10 k-fold validation for the rest of the algorithms. We have performed experiments with different neural network sizes in the same way as presented in [5] and decided that the hidden layer, for achieving the best results, should be composed of 7 neurons, according to Figure 6.

The results of metamodel training are presented in Table 3 as the maximum relative error expressed in percents. Training a single metamodel for the entire parameter space does not yield the best results. This fact can be observed for all evaluated machine learning algorithms. An advantage of modelling separate regions is combining distinct machine learning metamodels of distinct sub spaces for achieving best prediction results. In this case, R_{on} as a function of design parameters is most accurate approximated by GPR Matern

TABLE 3. *Technology B* Metamodel Evaluation [Maximum relative error %]

Algorithm	Single metamodel	Combined metamodel		
		M1	M2	Combined
Neural Networks	1.81	0.59	0.27	0.59
GPR Matern 5/2	2.40	0.50	0.42	0.50
GPR Rational Quadratic	2.91	0.70	0.60	0.70
SVM Cubic Regression	14.21	6.15	4.79	6.15
Linear Regression	82.03	41.23	25.41	41.23

5/2 for *M1* (the small part of sub space) and Neural Networks for *M2* (the large part of the sub space).

With the high accuracy metamodel achieved, the next step is the optimization task. As R_{on} has discontinuity points, using the same optimization algorithm as for *Technology A* does not lead to an optimal design. This aspect is illustrated in Figure 7 where it can be observed that optimal points (colored orange) resulting from the interior-point optimizer depend on the starting point. This figure was generated by running the optimizer 1000 times, each time with a different starting point for both the interior-point and differential

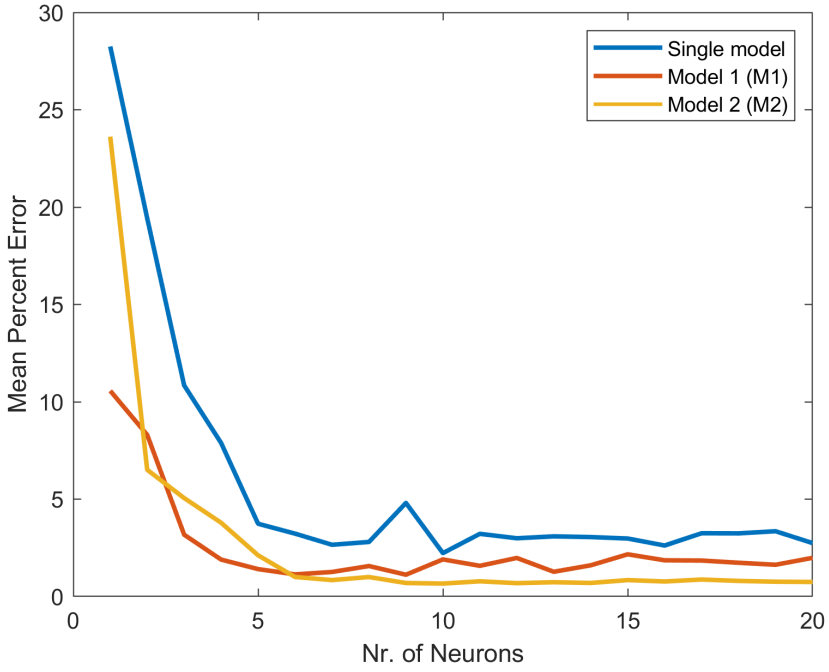

FIGURE 6. Choosing the size for Neural Network Model

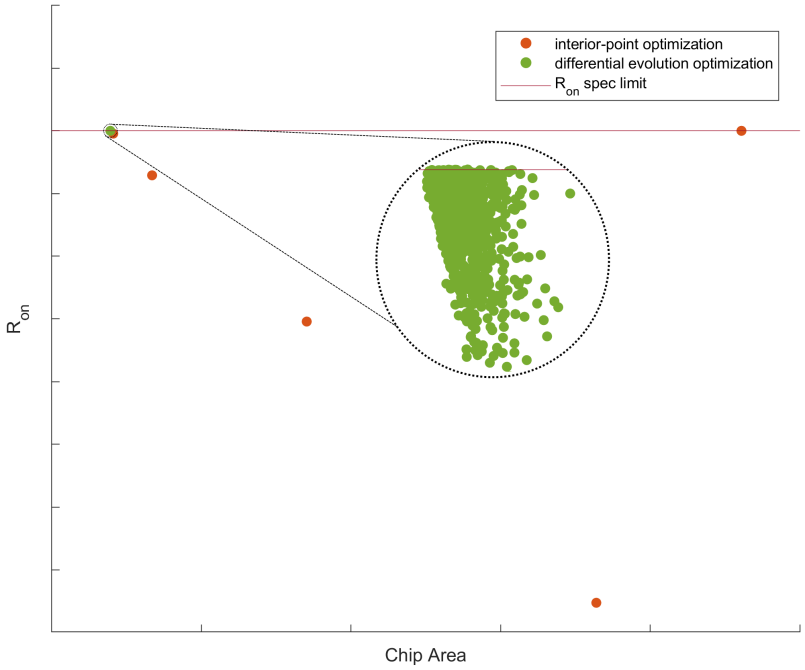

FIGURE 7. Chip Area optimization for *Technology B*

evolution optimizer. The optimal solutions of differential evolution are closer to the real optimum than the solutions yielded by interior-point optimizer. In this specific case, the worst-case solution of the interior-point optimizer has a 3.11% larger chip area than the worst-case solution of differential evolution.

7. Conclusion

This paper presents a methodology for metamodel-based optimization of MOSFET design. To achieve an automatic optimization, we propose a flow composed of 3 main blocks: Automated Simulation Setup, Prediction Metamodel and Design Optimizer. Several technical challenges of these blocks have been discussed together with related solutions. There are four main aspects when applying the current methodology for a technology:

- Automated Simulation Setup ensures results consistency;
- Simulation dataset covers design parameter space and its corners;
- Machine learning metamodels yield high accuracy;
- R_{on} function properties leads to optimization algorithm selection;

Two different package technology use cases have been detailed to illustrate that the methodology ensures optimization. The prediction metamodels yield very high accuracy, with the maximum relative error being less than 2% in both cases. The inference time of the machine learning metamodels is at most hundreds of μs while 3D finite simulation time is tens of minutes in the

presented use cases. Metamodels are fitted by performing an initial set of simulations for metamodel training. Additionally, optimization examples have been illustrated together with solution details to ensure design optimization for a required R_{on} while minimizing the area of the chip. When using metamodels for optimization, instead of using a classical simulation approach, the number of cost function evaluations becomes insignificant. The final results will be double-checked by 3D simulation. As future work, we consider developing complex metamodels that can predict other performance parameters and adding device technology parameters. In this way, complete device design optimization can be achieved automatically, or even technology itself can be improved through optimization.

REFERENCES

[1] M. R. Hontz, W. Collings, A. Courtay, and R. Khanna, "An optimization framework for gan power device design and applications," in *2019 IEEE 7th Workshop on Wide Bandgap Power Devices and Applications (WiPDA)*, pp. 302–309, 2019.

[2] H.-C. Choi, H. Yun, J.-S. Yoon, and R.-H. Baek, "Neural approach for modeling and optimizing si-mosfet manufacturing," *IEEE Access*, vol. 8, pp. 159351–159370, 2020.

[3] G. Bazzano, A. Raffa, S. A. Rizzo, N. Salerno, G. Susinni, and P. Veneziano, "Optimization of a sic mosfet behavioural circuit model by using a multi-objective genetic algorithm," in *2020 IEEE Energy Conversion Congress and Exposition (ECCE)*, pp. 2281–2286, 2020.

[4] G. Nicolae, C. Boianceanu, A. Buzo, C. Diaconu, H. Cucu, G. Pelz, and C. Burileanu, "Automatic parameter tuning in finite element analysis of semiconductor packages," in *2020 International Semiconductor Conference (CAS)*, pp. 41–44, 2020.

[5] G. Nicolae, A. Buzo, C. Feuerbaum, C. Diaconu, H. Cucu, G. Pelz, and C. Burileanu, "Metamodel-based prediction of on resistance for microelectronic power switches," in *IEEE Electrical Design of Advanced Packaging and Systems (EDAPS)*, pp. 1–3, 2021.

[6] X. Yan and X. G. Su, *Linear Regression Analysis: Theory and Computing*. USA: World Scientific Publishing Co., Inc., 2009.

[7] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," *Advances in Kernel Methods-Support Vector Learning*, vol. 208, 07 1998.

[8] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. USA: The MIT Press, 2006.

[9] H. Wang, B. Raj, and E. P. Xing, "On the origin of deep learning," *CoRR*, vol. abs/1702.07800, 2017.

[10] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999.

[11] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Berlin, Heidelberg: Springer-Verlag, 2005.