ISOP+: Machine Learning-Assisted Inverse Stack-Up Optimization for Advanced Package Design

Hyunsu Chae, Keren Zhu, *Member, IEEE*, Bhyrav Mutnury, *Fellow, IEEE*, Douglas E. Wallace Jr., Douglas S. Winterberg, Daniel de Araujo, *Senior Member, IEEE*, Jay Reddy, Adam Klivans, and David Z. Pan, *Fellow, IEEE*

***Abstract*—The future of computing requires heterogeneous integration, including the recent adoption of chiplet methodology, where high-speed cross-chip interconnects and packaging are critical for the overall system performance. As an example of advanced packaging, a high-density interconnect (HDI) printed circuit board (PCB) has been widely used in complex electronics ranging from cell phones to computing servers. A modern HDI PCB may have over 20 layers, each with its unique material properties and geometrical dimensions, i.e., stack-up, to meet various design constraints and performance requirements. Stack-up design is usually done manually in the industry, where experienced designers may devote many hours adjusting the physical dimensions and materials in order to meet the desired specifications. This process, however, is time-consuming, tedious, and suboptimal, largely depending on the designer's expertise. In this article, we propose to automate the stack-up design with a new framework, ISOP+, using machine learning (ML) for inverse stack-up optimization for advanced package design with adaptive weight adjustment and multilevel optimization. Given a target design specification, ISOP+ automatically searches for ideal stack-up design parameters while optimizing performance. A novel ML-assisted hyperparameter optimization method is developed to make the search efficient and reliable. Experimental results demonstrate that ISOP+ is better in figure-of-merit (FoM) than conventional simulated annealing and Bayesian optimization algorithms, with all our design targets met with a shorter runtime. We also compare our fully automated ISOP+ with expert designers in the industry and achieve very promising results, with orders of magnitude reduction of turn-around time.**

***Index Terms*—Design optimization, hyperparameter optimization, signal integrity (PCB stack-up, PCB interconnect design, packaging).**

Manuscript received 1 March 2023; revised 28 June 2023; accepted 2 August 2023. Date of publication 23 August 2023; date of current version 26 December 2023. This work was supported in part by NSF under Grant 1718570; in part by NSF through the AI Institute: Institute for Foundations of Machine Learning (IFML) under Grant 2019844; and in part by NSF AI Institute for Learning-Enabled Optimization at Scale (TILOS) under Grant 2112665. This article was recommended by Associate Editor C. Zhuo. *(Corresponding author: Hyunsu Chae.)*

Hyunsu Chae, Keren Zhu, and David Z. Pan are with the Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712 USA (e-mail: hyunsu.chae@utexas.edu; keren.zhu@utexas.edu; dpan@ece.utexas.edu).

Bhyrav Mutnury, Douglas E. Wallace Jr., and Douglas S. Winterberg are with Dell Technologies, Austin, TX, USA.

Daniel de Araujo is with Siemens EDA, Cedar Park, TX, USA.

Jay Reddy was with Dell Technologies, Austin, TX, USA. He is now with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712 USA.

Adam Klivans is with the Department of Computer Science, The University of Texas at Austin, Austin, TX 78712 USA.

Digital Object Identifier 10.1109/TCAD.2023.3305934

I. Introduction

ADVANCES in packaging technologies are driving the scaling of electronic devices. Advanced packaging technology involves integrating various materials and interfaces and requires collaboration between architects and engineers from multiple disciplines. Despite the slowing down of the integrated circuits (ICs) fabrication process, on-packaging interconnects and heterogeneous integration continues to propel the evolution of computing systems [1].

Following the trend, printed circuit board (PCB) technology is also evolving at a rapid pace. Modern high-performance PCB designs can typically have 12 to 20 layers [2]. Each layer contains a variety of signals, ranging from single-ended double data rate (DDR) signaling to differential serializer/deserializer (SerDes) routing. The developments of high-density interconnect (HDI) and substrate-like PCB (SLP) are also increasing the complexity of PCB stack-up design [3]. Advanced PCB, together with other trends in advanced packaging, is increasing the degree of integration between chiplets, where the signals on a PCB are sensitive to the physical design of a stack-up design. In a typical industrial design flow, the stack-up design is usually handled by experienced engineers manually through many trial-and-error and simulations to determine the choice of materials and their physical dimensions for each PCB layer. This manual process, which relies heavily on the intuition and experience of engineers, is critical for ensuring the signal integrity of PCB interconnects.

Automation of stack-up design can further optimize the interconnects. Historically, interconnect optimization in IC significantly contributes to the advances of the whole system [4], and a similar trend is observed in optimizing package-level interconnects for heterogeneous integration systems [1]. The high-speed PCB is one of the key components in the inter-chip interconnection, and its stack-up design significantly impacts the performance. However, the existing research in automating the PCB stack-up design is limited. Liao et al. [5] proposed to use an artificial neural network (ANN) to predict the resulting PCB performance, such as transmission line impedance, insertion loss, and cross-talk metrics. Then, a designer uses the ANN model to quickly obtain the stack-up design performance from PCB stack-up parameters without running time-consuming simulations and manually

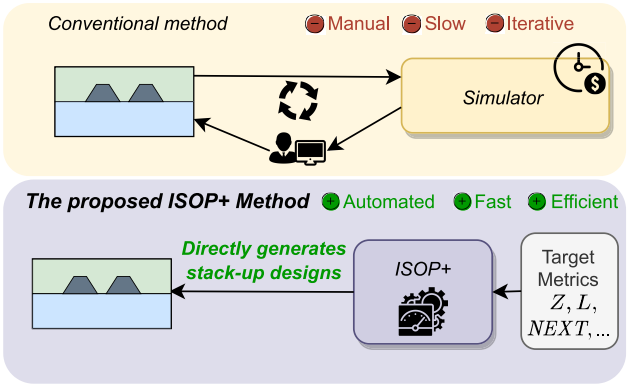


Fig. 1. Illustration of the conventional stack-up design and our proposed inverse stack-up optimization framework (ISOP+).

optimizing through iterations. He et al. [6] proposed using an integer programming-based method to generate stack-up arrangement candidates, which requires experienced package designers to define a set of appropriate design rules to accelerate the overall design cycle. Both [5] and [6] require extensive manual engineering efforts to make the design decisions. Kiguradze et al. [7] applied the Bayesian optimization (BO) method to optimize the five-parameter stack-up design. However, it only offers a solution for limited design space and fails to provide a fully automated and scalable solution to the stack-up designs.

In this article, we propose a fully automated stack-up design methodology in an inverse optimization setting, called inverse stack-up optimization for advanced package design (ISOP+). Instead of assisting the manual design, our proposed framework directly produces the stack-up designs that are optimized against performance requirements by leveraging automated search algorithms and machine learning (ML) surrogate model. As shown in Fig. 1, given the design specifications, such as transmission line impedance Z, ISOP+ searches for stack-up designs, and optimizes for performance metrics, such as signal loss at various frequencies and cross-talk. The main contributions of this work are summarized as follows.

1) We propose ISOP+, an agile framework for automating the HDI PCB stack-up design process. To the best of our knowledge, this is the first work to provide a fully automated solution to stack-up design. We believe the proposed methodology, in general, can be extended to many other scenarios of interconnect optimization.
2) We formulate the inverse stack-up optimization as a hyperparameter optimization (HPO) problem, using the customized optimization flow that incorporates both global and local search. The global stage is powered by an effective two-stage HPO search algorithm to solve the stack-up design by efficiently and quickly exploring the entire search space. The local stage leverages available gradient information for further refinement.
3) We accelerate the optimization process by introducing ML surrogate models to predict stack-up design performance. The ML models replace the expensive simulations in the search for space exploration.
4) We enhance the quality and practicality of the optimization outcome by incorporating adaptive weight adjustment and input parameter constraints into the optimization objective.
5) Experimental results demonstrate that the ISOP+ framework reduces the design cycle from hours to minutes and greatly outperforms other baseline optimization methods as well as the manual design.

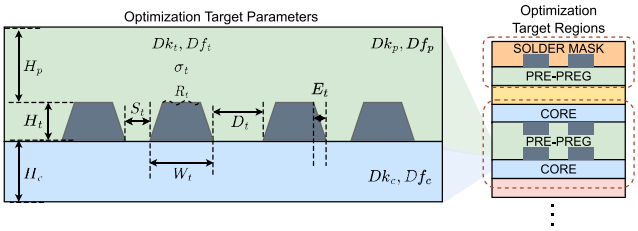


Fig. 2. Structure of a differential stripline layer.

The remainder of this article is organized as follows. We introduce the stack-up design problem and its formulation in Section II. The details of the proposed framework and its key elements are described in Section III. Section IV presents the experimental results. The conclusions are provided in Section V.

II. Preliminaries

In this section, we introduce the preliminaries of PCB stack-up design (Section II-A) and HPO problem (Section II-B). We then formulate the inverse stack-up design problem (Section II-C).

A. PCB Stack-Up Design

Construction of a PCB begins with designing its material layers' arrangement, i.e., stack-up. The basic function of a PCB, which comprises primarily of passive interconnects, is to transfer a signal from one port to another while maintaining signal integrity. Since establishing a stack-up is the first step in PCB design, it has a significant impact on the efficiency and cost of the final product. In high-speed systems, a well-designed stack-up can lessen the circuit's susceptibility to external noise and cross-talk issues. The performance of a transmission line is determined by a layer's physical dimensions and material properties. A simplified structure of a single differential stripline layer is illustrated in Fig. 2. The subscript t, c, and p denote a layer's metal trace for signals, glass-reinforced epoxy laminate sheet (core), and preimpregnated bonding sheet (prepreg), respectively. The notations of each parameter are presented Table I.

In conventional industrial design flow, a designer usually selects a combination of design parameters and uses a computationally expensive electromagnetic (EM) field simulation software to evaluate the selection [8]. An experienced designer has an understanding of the physics of transmission lines, and how the parameters affect the system's individual performance. For example, the impedance of the transmission line is the root of inductance over capacitance. As the metal trace width increases, the capacitance of the transmission line increases, leading to a decrease in impedance. However, it is hard to see the second-order effect of these parameters, and how it exactly affects the multiple system requirements. Software EM solver

TABLE I
NOTATIONS OF STACK-UP PARAMETERS

Physical Dimensions	
W	width
H	height
S	spacing between differential signals
D	distance between two differential pairs
E	etch factor (represent the trapezoidal shape)

Material Properties	
Df	dissipation factor
Dk	dielectric constant
σ	conductivity
R	surface roughness

tool based on integrated channel analysis tool (ICAT) [9] is one example. It takes about 1 min to compute performance metrics for each layer of the stack-up design, and varies depending on the machine resources at hand. The designer evaluates the simulation result against the system's requirements, which include matching differential impedance, minimizing insertion loss, and minimizing near-end and far-end cross-talk. Optimization of stack-up design is a time-consuming task that requires a number of iterations of simulations by trial-and-error. Furthermore, designers' reliance on heuristics and intuition can cause them to overlook nonintuitive solutions. Our studies show that manual stack-up designs often result in inferior quality, especially when facing tradeoffs between different performance metrics. In this work, we propose an automated and efficient stack-up design framework that can significantly reduce manual efforts and turnaround time while producing improved solution quality.

B. Hyperparameter Optimization

HPO searches for the best set of parameters in an optimization problem that is costly to evaluate in general. In general, an HPO aims to obtain a parameter set $\mathbf{x}$ as shown in

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \tag{1}$$

where $\mathbf{x}$ denotes the hyperparameters and $f(\mathbf{x})$ is the objective function to be minimized. Unlike traditional optimization problems, the objective functions in HPO are usually nonconvex and nondifferentiable, which blocks the adoption of many optimization techniques. Meanwhile, the evaluation time for $f(\mathbf{x})$ is often non-negligible, which imposes a higher requirement on sampling efficiency on HPO algorithms.

There have been several existing methods that target the HPO problems, including [10]. Grid search and random search are two simple approaches. Their search schemes do not leverage the evaluated records, so the sample efficiency is usually low. Metaheuristic algorithms, such as genetic algorithms and simulated annealing (SA), use a heuristic algorithm to guide the randomized search process to improve efficiency. In recent years, BO and its variants have become popular with HPO [11]. BO is an iterative process that builds a surrogate model to fit observed points into the objective function and guides the exploration. An acquisition function is used to balance exploration and exploitation. BO is often efficient but hard to parallelize due to its sequential nature.

HPO has been used to tune parameters in design flow for very large scale integration (VLSI) [12], [13], [14], [15], [16] and field-programmable gate array (FPGA) [17]. It can also be used to optimize the hyperparameters for individual stages, such as placement [18], [19]. Besides tuning the parameters, HPO is also adopted in solving the analog device sizing problem. The automated analog sizing methods work on the inverse design problem. Given target specifications, automatic analog sizing treats design parameters, such as transistor width, as hyperparameters and applies HPO to find the sizing solution. There have been a variety of HPO search algorithms applied to the analog sizing problem, including the genetic algorithm [20], BO [21], and reinforcement learning [22]. Inspired by the analog sizing problem, we apply HPO to automate the stack-up design in inverse optimization.

C. Inverse PCB Stack-Up Optimization: Problem Formulation

The inverse PCB stack-up optimization process searches for a set of design parameters that meet the system specifications while optimizing a user-defined figure of merits (FoMs). In science and mathematics, an inverse problem often refers to estimating the unknown parameters inversely through measurements. Similarly, our optimization scheme seeks to identify valid stack-up design parameters by obtaining information about the target performance measurements.

Problem 1 (Inverse PCB Stack-Up Optimization): Given a set of input search range $\mathcal{S}$ and a set of performance constraints C, solve the optimization problem and obtain the stack-up design parameters as shown in

$$\begin{aligned} \mathbf{x}^* = \arg\min_{\mathbf{x}} \; & f^{\text{FoM}}(\mathbf{x}) \\ \text{subject to} \; & x_i \in \mathcal{S}_i \text{ for } i = 1, \ldots, d \\ & f_j^C(\mathbf{x}) \le 0 \text{ for } j = 1, \ldots, k \end{aligned} \tag{2}$$

where $\mathbf{x}$ is a d-dimensional parameter vector, and $\mathcal{S}_i$ denotes the set of valid numbers for parameter x_i. f^{FoM} is the FoM function to optimize, $\mathcal{S}$ defines the input search space, and f^C denotes the performance specifications constraints. k is the number of constraints in the problem.

III. ALGORITHMS

This section presents our ISOP+ framework and the optimization of the inverse stack-up design. It shall be noted that our framework is easily extensible to other advanced packaging designs that require stack-up design and optimizations.

A. Overall Flow

The ISOP+ offers an automated and effective framework for inverse stack-up design optimization tasks. The goal of the inverse stack-up design optimization is to determine the best combination of design parameters for each layer in a PCB's stack-up. The optimized final design must meet a specified performance metric, as well as comply with performance specifications and constraints. The evaluation of both performance metric and constraints are nontrivial.

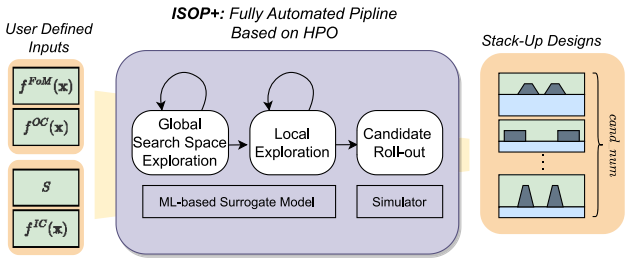


Fig. 3. Overall flow of ISOP+ framework.

Traditional manual design flow relies on an engineer's experience and trial-and-error approach using slow simulations to optimize a stack-up design. ISOP+ provides a faster, more reliable, and more practical alternative.

The ISOP+ framework solves the inverse PCB stack-up optimization by incorporating a discrete domain HPO and gradient descent. Fig. 3 shows an illustration of the overall flow. It takes four categories of inputs, an FoM function (f^{FoM}), a set of performance output constraints (f^{OC}), a set of parameter search spaces ($\mathcal{S}$), and a set of design parameter input constraints (f^{IC}). f^{FoM} and f^{OC} are related to the performance metrics, and $\mathcal{S}$ and f^{IC} are related to the input design parameters. The output of ISOP+ is the design parameters for stack-up design. The optimization process contains three stages: 1) early global search exploration; 2) local exploration; and 3) later candidate roll-out. The first stage samples the parameters globally to explore the search space and shrink it. We utilize a discrete domain HPO method and ML surrogate model to effectively explore the different local minima in this stage. The second stage selects a small number of samples from the constrained search space from the previous stage and refines them through local optimization. This stage utilizes the gradient approximation available through the ML surrogate model and employs gradient descent to the samples, allowing for quick fine-tuning of multiple local minima. The last stage of candidate roll-out then chooses the final optimal stack-up designs.

Algorithm 1 illustrates the procedures of our ISOP+ framework in detail.

1) *Global Search Space Exploration (Lines 1–8):* Our stack-up problem is a nonconvex optimization that consists of multiple local minima solutions rather than a single global minimum. As a result, it is crucial to comprehensively investigate various favorable points throughout the entire design space during the global optimization phase. First, we encode the initial search space and input parameter values to binary values (lines 1 and 2). Then, the search space is iteratively reduced to eliminate stack-up parameter space that is not ideal (lines 3–7). The optimization process is guided by the optimization objective function $\hat{g}(\cdot)$, which is devised to incorporate f^{FoM}, f^{OC}, and f^{IC}. Furthermore, to facilitate the search space exploration, $\hat{g}(\cdot)$ is adaptively adjusted as the search space narrows (line 6). We intend to obtain more samples and rapidly reduce the search space in a tradeoff of some accuracy. Instead of conducting time-consuming EM simulations, the performance metrics are estimated by sampling from an ML surrogate model $\hat{M}(\cdot)$ to evaluate $\hat{g}(\cdot)$. Finally, several samples are selected from the reduced search space (line 8).
2) *Local Exploration (Lines 9–12):* During this stage, the solution candidates from the previous stage are further fine-tuned, taking advantage of the gradient approximation provided by the ML surrogate model. This gradient descent-based method excels at exploring neighboring points that offer improvements over the current solution with fast speed. To begin this stage, we decode our binary search space and design parameter candidates to decimal domain to apply the gradient descent algorithm on continuous domain (line 9). We employ the selected samples as initial points for stochastic gradient descent-based local optimization using the Adam optimization algorithm, also employing the ML surrogate model (lines 10–12).
3) *Candidate Roll-Out (Line 13–16):* After local refinement of the search space, we roll out *cand_num* design candidates. In this stage, we ensure that design parameters are valid by rounding to the nearest valid discrete values (line 14). Then, we further evaluate them with accurate EM simulations ($M(\cdot)$) and choose the final solution based on an objective function $g(\cdot)$ (line 15). Function $g(\cdot)$ is devised to incorporate our ultimate optimization goal, f^{FoM} and f^{OC}. ISOP+ can generate multiple design candidates (*cand_num*) ranked by FoM in the roll-out stage if specified by the user.

Algorithm 1 ISOP+

Input: $f^{\text{FoM}}(\cdot)$, $f^{IC}(\cdot)$, $f^{\text{OC}}(\cdot)$, $\mathcal{S}$
Output: *cand_num* combination of $\mathbf{x}^*$

1: `Encode` $\mathbf{x}$ and $\mathcal{S}$ to binary domain
2: search space $T(\mathbf{x}) \leftarrow \mathcal{S}$
3: **for** $i \leftarrow 1$ to *iter_num* **do**
4: Take q random sample $\mathbf{x}^q$ from $T(\mathbf{x})$
5: $T(\mathbf{x}) \leftarrow$ `UpdateSpace`$(T(\mathbf{x}), \mathbf{x}^q, \hat{g}(\hat{M}(\cdot)))$
6: $\hat{g}(\cdot) \leftarrow$ `UpdateWeights`$(\mathbf{x}^q, \hat{g}(\cdot))$
7: **end for**
8: Take p sample $\mathbf{x}^p$ from $T(\mathbf{x})$ using Hyperband
9: `Decode` $\mathbf{x}$ and $\mathcal{S}$ to decimal domain
10: **for** $i \leftarrow 1$ to *epoch_num* **do**
11: $\mathbf{x}^p \leftarrow$ `AdamOptimizer`$(\mathbf{x}^p, \hat{g}(\hat{M}(\cdot)))$
12: **end for**
13: **for** $i \leftarrow 1$ to *cand_num* **do**
14: $\mathbf{x}_i \leftarrow$ `RoundToValidDiscreteValue`$(\mathbf{x}^p)$
15: $\mathbf{y}_i \leftarrow g(M(\mathbf{x}_i))$
16: **end for**

1) 2) 3)

The rest of the section presents details on the ISOP+ framework algorithm's key components.

B. HPO Search Algorithm—Global Exploration

ISOP+ adopts and adapts the Harmonica algorithm [23] in the exploration of the global search space. By leveraging Harmonica, the parallelized evaluation of design configurations is possible, and we can ensure the optimization

task remains within the constrained discrete search space. Harmonica is a spectral approach to discrete domain HPO that is predicated on the notion that the objective function can be represented by a combination of sparse and low-degree Fourier polynomials. The algorithm utilizes the polynomial sparse recovery (PSR) subroutine to gradually reduce the search space. Equation (3) represents a simplified version of Harmonica's PSR subroutine

$$p(\mathbf{x}) = \sum \alpha_{c_i} \psi_{c_i}(\mathbf{x}) \tag{3}$$
$$\text{subject to} \quad \arg\min_{\alpha} \left\{ \sum_{i=1}^{q} \left(\sum \alpha_c \psi_c(\mathbf{x}_i) - \mathcal{F}(\mathbf{x}_i) \right)^2 \right\}$$

where $c_1, \ldots, c_k$ is the indices of the most significant k components of the Fourier coefficients α, and $\mathcal{F}(\cdot)$ denotes the objective function, which is $\hat{g}(\cdot)$ in our task. $\mathbf{x}$ is binary vector of size n represented in $\{-1, 1\}^n$. This approximation method can efficiently reconstruct the objective function by observing limited randomized samples with several iterations [24]. The Harmonica algorithm observes q random samples of $\mathcal{F}(\mathbf{x}_i)$ in each iteration. By solving sparse polynomial regression using PSR subroutine, it chooses the significant bits for $\mathbf{x}^*$ based on a polynomial approximation $p(\mathbf{x})$. Each iteration selects a random q set of variable combinations $\mathbf{x_i}$ from the reduced search space and further evaluates them with the performance models.

The proposed HPO search algorithm allows for efficient parallelized sampling and evaluation of design parameters in the ISOP+ framework. In each iteration, a large number of random sets of design parameters are sampled over the search space in batch. These candidate parameter sets are evaluated in parallel by the ISOP+ framework using the objective function. Then, they are incorporated into a simple linear regression method to approximate a polynomial to fit the optimization objective function. In comparison to typical sequential HPO algorithms, such as BO, the suggested parallelized method, enables us to observe and obtain more samples within the same runtime budget, resulting in improved optimization outcomes. This parallelized nature of the method enables faster convergence and is beneficial for the task with a very large initial search space.

The Harmonica algorithm can in-situ optimize over a constrained and discrete parameter search space. A discrete domain consists of a finite or countable number of values, while a continuous domain consists of an uncountable number of real values. When the optimization task is carried out in a discrete domain, there are fewer possible solutions to consider, which can simplify the search for the global optimum. In addition, in the context of stack-up design, design parameters are constrained to discrete values and specific ranges due to material and technology selection limitations. Therefore, by assuming a discrete search space for our problem, we can ensure that no loss of information happens during the optimization process. The granularity or resolution of each parameter can be incorporated as parameter increment step size. In ISOP+, we translate the entire discrete search space $\mathcal{S}$ to a binary search space $T(\mathbf{x})$, and we directly choose the design configuration from this encoded space. For example, for a decimal representation of a design parameter $x_{(10)}$, the binary domain representation $x_{(2)}$ will be as shown in

$$x_{(2)} = \texttt{ToBinary}\left(\frac{x_{(10)} - x_L}{dx} \right) \tag{4}$$

where $[x_L, x_U]$ is the defined parameter range and dx is the parameter increment size. The number of bits assigned for each parameters can be represented as $\log_2([(x_U - x_L)/dx] + 1)$, and this information is used to map each design choices to a binary value. In binary space, each parameter is represented by a limited number of bits; therefore, the variables are certain to be within the search space $\mathcal{S}$. The such property benefits the overall efficiency of the global optimization process by avoiding explorations with invalid design parameter values.

In our implementation, the final iteration of the Harmonica algorithm is followed by the hyperband algorithm [25] to select several design parameter candidates for further evaluation in the local optimization phase. Hyperband is a bandit algorithm that balances random exploration versus informed exploitation when given a limited sampling budget B. Compared to the BO approach, it is fast in speed by doing random searches through adaptive resource allocation and early stopping. It randomly selects a set of parameter configurations, n, and evaluates them using different resource budgets. In the iterative process, the algorithm discards the worst-several cases and allocates more resources in the promising configuration set. If allocating the uniform budget B/n over all configurations, deciding n becomes a problem of tradeoffs. The hyperband algorithm adaptively allocates resources by considering several possible values of n for a fixed B, in a grid search-manner. Our experiments have shown that this sampling method outperforms the naive random sampling approach. The chosen samples from this global optimization are then optimized further in the subsequent optimization procedure.

C. Gradient Descent—Local Exploration

We optimize further by performing gradient descent on selected solution candidates. The global optimization process aims to reduce the search space and find an approximate solution close to the true optimal. The local optimization task then seeks to refine the solution candidates by examining samples in the vicinity of each solution candidate. In Harmonica's step, encoding each possible parameter value as binary can be beneficial for optimization tasks to stay within the constrained design space, especially since the sizeable initial space is considered. However, this can pose a limitation when working with values close to the boundaries of the binary domain. For example, in a continuous domain, there is a difference of 1 between the values 31 and 32. However, it will appear as all bits flipped in their 5-bit binary representation (5'b01111 and 5'b10000). Hence, it is necessary to inspect the sample points further to optimize locally in a continuous domain.

We need to consider conversions between discrete and continuous domains to utilize gradient descent. First, prior to the gradient descent stage, solution candidates are converted back to decimal form (Algorithm 1 line 9), following:

$$x_{(10)} = \texttt{ToDecimal}\left(x_{(2)}\right) \times dx + x_L. \tag{5}$$

Then, once the gradient descent process is complete, we round the parameter values to the nearest discrete values (Algorithm 1 line 14) as shown in

$$\text{final } x = dx \times \texttt{Round}\left(\frac{x_{(10)}}{dx}\right). \tag{6}$$

This process ensures that our final solution configurations are valid discrete values. While this might cause some loss in accuracy, it is negligible in the context of local search. We have the ability to control the learning rate and epoch number of the gradient descent algorithm. This ensures that the optimization process does not excessively fine-tune the parameters to the extent that the changes become insignificant when rounded to the nearest discrete values.

A stochastic gradient descent-based Adam optimizer [26] is employed to refine the candidate samples. Adam optimizer is a widely used optimization algorithm for training deep learning models for its ability to reach good results at a fast speed. To construct an optimizer that is suitable for our problem, we directly input our design parameter $\mathbf{x}$ into the optimizer. This enables the optimizer to compute the gradient for each design parameter and adjust them in a way that minimizes the loss. Additionally, we include the typical hyperparameters for the gradient descent algorithm, such as learning rate. The p best-candidate samples obtained from the previous global optimization stage serve as the initial point of one batch. We use the trained ML surrogate model and the objective function $\hat{g}(\cdot)$ as loss function to evaluate the samples. The Adam optimizer calculates the gradient and attempts to improve the average loss of the whole batch. The local optimization stage concludes by selecting *cand_num* design configurations for the candidate roll-out stage, where they undergo evaluation using real EM simulations.

D. ML-Based Surrogate Models

The ML-based surrogate model is used to replace the expensive EM simulation in the exploration stage to accelerate the optimization process. The optimization procedure requires frequent performance evaluations of different design parameters. Relying on the slow EM simulations makes the HPO process and gradient descent slow and limits the number of samples we can afford. To address this issue, we use a fast ML-based surrogate model to increase the number of samples the search algorithm can observe within the available runtime budget.

Building our ML surrogate models is essentially a regression problem with tabular features. Our dataset contains 90k unique stack-up design combinations, of which 80% are used for training and 20% for testing. The data is collected using an industry-standard simulation tool based on ICAT [9]. We randomly query the data over the wide range of each parameter set by the designers. The design parameters and ranges represent a large solution space, 10^{29}. While the training dataset is equivalent to only $7 \times 10^{-23}\%$ of the entire search space, our ML surrogate model empirically results in satisfying accuracy and can guide our search space exploration effectively.

We select the structure of 1D-convolutional neural network (CNN) model [27] for our surrogate model. Conventionally, CNN is used for image and video processing tasks due to its efficiency in capturing local input patterns by considering spatial structure associativity of the data. Suppose a CNN is applied to a tabular dataset. In that case, it assumes a correlation between each tabular feature, and the ordering of the features impacts the model accuracy. However, it is hard to define the correct locality correlation of features in a tabular dataset, such as our stack-up dataset. 1D-CNN suggests the preprocessing layer of reordering and reshaping to find the correlation of the features before processing to the CNN layers, as shown in Fig. 4. Before the typical convolutional layers are initiated, the tabular features are passed through a fully connected layer. This procedure generates the new features as a linear sum of numerous original features, expanding the feature dimension. Then, the feature vector is reshaped to form the 3-D image format. In this manner, signals with a large number of possible orderings of distinct characteristics are generated before passing through the fundamental CNN model. Consequently, we utilize the CNN structure by intentionally generating a signal suited for CNN.

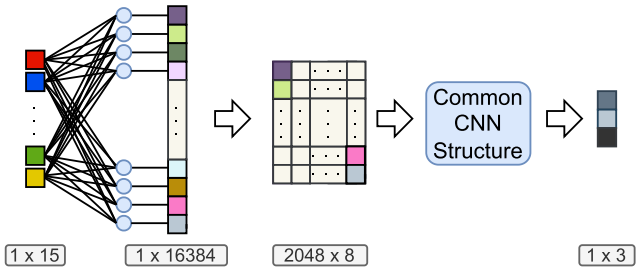


Fig. 4. Simplified structure of 1D-CNN [27].

Utilizing 1D-CNN, we train one multioutput model to predict impedance, signal loss, and near-ended cross-talk. By using a single multioutput model rather than constructing individual models for each, we allow outputs to share information and have complex interactions that can only be handled by structured inference. Also, it is computationally more efficient than training and maintaining multiple single models, providing unified prediction rules and reducing the training time. We utilize the model as a proxy; therefore, it must produce accurate predictions within an error margin relative to the actual value. Mean average percentage error (MAPE) is used as the primary evaluation metric to train the models for impedance and loss, and symmetric mean absolute percentage error (sMAPE) for cross-talk as it could have zero values. There are a variety of regression methods, including XGBoost [28], decision tree regressor (DTR), gradient boosting regressor (GBR), polynomial linear regression (PLR), random forest regressor (RFR), support vector regressor (SVR), and multilayer perceptron regressor (MLPR) [29]. The comparison study of the different models and their accuracy is presented in Section IV-B.

E. Optimization Objective Function

We introduce an objective function $\hat{g}(\cdot)$ for optimization in the early global search space exploration stage. The inverse stack-up optimization problem is to minimize f^{FoM} while

honoring the constraint f^{IC} and f^{OC}. However, directly solving the constraint problem is difficult and inefficient. Evaluation of f^{FoM} and f^{OC} requires querying from performance models, which requires computational resources. A naive approach ignores the constraints in the exploration stage and filters out the invalid results at the last stage. This blind search space exploration could result in unnecessary computation that violates the constraint and produces a poor-quality solution.

In this work, we propose to relax the constraints into the penalties in the optimization objective function, $g(\cdot)$, to guide our optimization task appropriately. Consequently, our problem statement becomes

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} g(\mathbf{x}) \tag{7}$$

$$g(\mathbf{x}) = \sum_i w_i^{\text{FoM}} \cdot f_i^{\text{FoM}}(\mathbf{x}) + \sum_j w_j^{\text{OC}} \cdot f_j^{\text{OC}} \tag{8}$$

where $f_j^{\text{OC}}(\mathbf{x}) = \max\big(M_j(\mathbf{x}) - f_{j\pm}, 0\big)$.

The f_j^{OC} becomes a clip function for constraints. For instance, one may wish to give a constraint for impedance such that it has an acceptable tolerance range, $Z_\pm$, of the characteristic impedance Z_o. The constraint function will become $f_Z^C(\mathbf{x}) = \max(|M_Z(\mathbf{x}) - Z_o| - Z_\pm, 0)$. Note that the input parameter constraint f^{IC} will be discussed further in the next section.

We further smooth $g(\cdot)$ into our objective function $\hat{g}(\cdot)$. In the HPO search scheme, the Harmonica algorithm approximates a polynomial function to guide the search space reduction. Directly modeling the nondifferentiable function $g(\cdot)$ is empirically inefficient and inaccurate. A smooth objective function would enable more searches at the border. Therefore, we enhance the performance of our HPO algorithm and locate local and global optimal points more efficiently. We suggest a smoothed approximation of the maximum function using the double sigmoid functions, $\hat{g}(\mathbf{x})$. $S(\cdot)$ indicates a sigmoid function

$$\hat{g}(\mathbf{x}) = \sum_i w_i^{\text{FoM}} \cdot f_i^{\text{FoM}}(\mathbf{x}) + \sum_j w_j^{\text{OC}} \cdot \hat{f}_j^{\text{OC}}(\mathbf{x})$$

where

$$\hat{f}_j^{\text{OC}}(\mathbf{x}) = S\Big(\gamma_j \cdot \hat{M}_j(\mathbf{x}) - f_{j\pm}\Big) + S\Big(-\gamma_j \cdot \hat{M}_j(\mathbf{x}) - f_{j\pm}\Big). \tag{9}$$

Our framework utilizes both $g(\cdot)$ and $\hat{g}(\cdot)$. Fig. 5 gives a comparison between the two. We can adjust $\hat{g}(\cdot)$ furthermore by giving control parameter γ. While $\hat{g}(\cdot)$ is utilized for the ML surrogate model to help better navigate the search space, $g(\cdot)$ is used in the later roll-out simulation stage, as this is our ultimate objective. The parameter γ influences the steepness of the constraint factor, as demonstrated in Fig. 5. When the tolerance factor $f_{j\pm}$ is of a tight region, we want our slope to be steep so that the boundary between values within and outside the restriction is more easily distinguishable. Therefore, we empirically determine the control parameter γ to be $1/f_{j\pm}$ in the experiments.

F. Input Parameter Constraint

Input parameter constraint f^{IC} is applied to establish interdependence between parameters and constrict the parameter space further. Another consideration when optimizing a design is to see the effect when given a limited physical design space. For instance, we may want to restrict the total horizontal length of a differential stripline. If no input parameter constraints are imposed, we will adjust the range of individual parameters, such as W_t, D_t, and S_t. However, this results in a manually limited search space, and the optimization task will have to find the best solution within this space rather than considering the compromise between parameters. Therefore, by adding the input parameter constraint to objective function $\hat{g}(\cdot)$, we can achieve greater control over the optimized design.

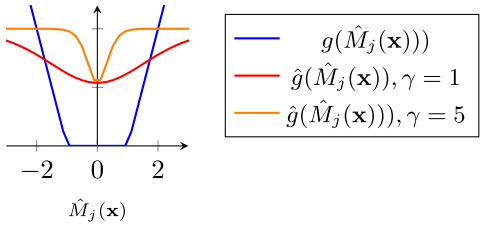


Fig. 5. Functions $g(\cdot)$ and $\hat{g}(\cdot)$ with different γ.

We combine the input parameter constraint f^{IC} to the objective function $\hat{g}(\cdot)$ as shown in

$$\hat{g}(\mathbf{x}) = \sum_i w_i^{\text{FoM}} \cdot f_i^{\text{FoM}}(\mathbf{x}) + \sum_j w_j^{\text{OC}} \cdot \hat{f}_j^{\text{OC}} + \sum_k w_k^{\text{IC}} \cdot f_k^{\text{IC}}(\mathbf{x}). \tag{10}$$

The f^{IC} term can be represented as a clip function

$$f_k^{\text{IC}}(\mathbf{x}) = \max(y_k(\mathbf{x}) - A_k, 0) \quad \text{where the constraint is } y_k(\mathbf{x}) \le A_k. \tag{11}$$

The function $y_k(\mathbf{x})$ is a first-order polynomial of $\mathbf{x}$. For example, to establish an upper limit A_1 for the horizontal dimensions W_t and S_t of the stack-up design, the following equation can be used: $y_1(\mathbf{x}) = 2 \times W_t + S_t$. The optimization process is controlled through the use of clip functions which make the design go against the direction of violation as it increases. This input constraint does not need to be evaluated by an ML surrogate model or simulation, and it does not require a smooth function, unlike the output constraint f^{OC}. The outcome for the objective function when the input constraint is incorporated can be found in Section IV-D.

G. Adaptive Weight Adjustment

We actively tune the weight w^{IC} and w^{OC} with respect to weight w^{FoM} to help guide the optimization task more effectively. Incorrect weight settings may result in poor optimization. If the constraint factors are weighted too high, it may hinder the efficient exploration of the area that best optimizes our ultimate objective. Alternatively, if the constraint factors are given insufficient weight, we may obtain meaningless results that violate the constraints. As the optimization process progresses, weight factors must be modified to account for the shrinking search space. Therefore, it is necessary to use the information gained from each iteration of the HPO

Algorithm 2 `update_weights()`

1: constraint_valid_ratio $= \frac{\sum(f^C \le wC_{\max})}{total\_sample\_num}$
2: **if** constraint_valid_ratio $\ge \beta$ **then**
3: $\quad w \leftarrow \max((1-\beta)w, \ \frac{\min(w^{\text{FoM}} \times \text{FoM}(\mathbf{x}))}{f} C_{\max})$
4: **end if**

search process to actively adjust and balance the weight of each factor.

Algorithm 2 presents the weight adjustment algorithm. Our HPO algorithm provides a batch of a new sample every iteration, allowing us to observe these random samples' statistics to help navigate the weight adjustment. First, we determine the proportion of randomly gathered samples that satisfy the constraint (line 1). When we determine that the ratio β is sufficient, we reduce the weights for the constraint factor (lines 2–4). However, there is a lower limit to the weight; if there is no lower bound, the weight could vanish, and the constraint could be ignored. Consequently, we use a clip function in which the minimum weight cannot be less than the minimum of FoM term. $C_{\max}$ represents the boundary condition of the acceptable regions for $\hat{f}^{\text{OC}}(\mathbf{x})$ and $f^{\text{IC}}(\mathbf{x})$. $C_{\max}$ is $\hat{f}^{\text{OC}}(-f_{\pm}/\gamma)$ and $\hat{f}^{\text{IC}}(0)$, respectively, for output and input constraints. In the experiment, we empirically choose β as 0.2. The output constraint factor $\hat{f}^{\text{OC}}(\mathbf{x})$ is guaranteed to be $(0, 2)$ because they are represented as a sum of two sigmoids. Also, our f^{FoM} term, which is mostly L for our experiments, is desirable to be $(-1, 0]$. Therefore, we empirically choose the initial weight of each factor, w^{FoM}, w^{OC}, and w^{IC}, as the same. The active weight adjusting continues throughout the HPO process, and the final weight values are utilized as fixed values in the subsequent gradient descent stage.

IV. Experimental Results

In this section, we present experimental results for the ISOP framework to prove its effectiveness and reliability. All the algorithms, the proposed and baselines, are implemented in Python and executed on a Linux machine equipped with GPU support. The final performance measurements are validated using EM simulations on a Windows machine. We first evaluate the effectiveness of the proposed inverse optimization method in Section IV-A in comparison to the baseline methods. We employ the same ML surrogate model across all methods to focus on evaluating the impact of different methods on overall optimization performance. Section IV-B provides the accuracy evaluation of our ML-based surrogate model. Section IV-C gives the comparative analysis of the various versions of ISOP. The method ISOP [30] differ from the proposed method in two key aspects, 1) the overall optimization techniques and 2) the ML surrogate model. This section highlights the specific contribution of ISOP+ in these two aspects to attaining positive improvements in overall performance. Finally, Section IV-D contains a comparative study between the ISOP-generated and the manual stack-up design.

TABLE II
Description of Each Experiment Tasks

Task	f^{FoM} / f^{OC}	Z_o (Ω)	$Z_{\pm}$ (Ω)	$NEXT_o$ (mV)	$NEXT_{\pm}$ (mV)
T1	$\{L\}/\{Z\}$	85	1	-	-
T2	$\{L\}/\{Z\}$	100	2	-	-
T3	$\{L\}/\{Z,\ NEXT\}$	85	1	0	0.05
T4	$\{L+2\cdot NEXT\}/\{Z\}$	85	1	-	-

A. Evaluation of the HPO Framework

This section presents the experimental results of the comparison between the proposed framework and other optimization techniques, including SA and BO. All techniques employ the same ML surrogate model to evaluate the performance of the solution candidate samples in the global and local optimization stage. Additionally, we use objective function $\hat{g}(\cdot)$ with the same initial weights. We further adjust the run setting of the SA and BO to match either the runtime (denoted as "-1") or the number of observed samples (denoted as "-2") of the proposed ISOP+. For example, SA-1 refers to the SA algorithm that has a runtime comparable to that of ISOP+, while SA-2 refers to a scenario where the number of samples observed is comparable to that of ISOP+. All tasks are terminated after 1000 s, and the results are reported at the termination. The BO algorithm is slow in the experiments due to its sequential process, and BO-2 experiments are terminated earlier.

Our BO implementation utilizes Optuna [31], which provides a simple and efficient implementation of BO. Optuna focuses on the tree-structured Parzen estimators (TPEs) algorithm instead of relying on Gaussian processes as its underlying surrogate model. It uses kernel density estimations to model the probability distribution of the objective function, making it computationally efficient for high-dimensional optimization problems. To maximize the number of BO iterations to achieve high-quality results, we adopt a sequential approach by evaluating a single sample each iteration. For SA implementation, we implement our own. The algorithm begins by randomly selecting an initial solution and then explores the search space iteratively by randomly selecting neighboring solutions. If a neighboring solution proves to be better than the current solution, it is accepted as the new solution. However, if the neighboring solution is worse, it can still be accepted with a certain probability. This allows the algorithm to explore different search space regions and increases the chance of finding the global optimum. In our implementation, this probability is decided by comparing exp([cost − new_cost]/temperature) and randomly generated number between [0.0, 1.0). The temperature decreases linearly as the iteration progresses.

The experiment conducts optimization tasks with four different user objectives (FoM) and constraints as described in Table II. Z is differential impedance, $|Z_o|$ is the transmission line's characteristic differential impedance, and $Z_{\pm}$ is the acceptable tolerance of $|Z - Z_o|$. L is differential insertion loss at 16 GHz in dB/inch. $NEXT$ is the peak near-end differential cross-talk in mV. $|NEXT_o|$ and $NEXT_{\pm} = |NEXT - NEXT_o|$ are the target and acceptable tolerance, respectively, when $NEXT$

TABLE III
DESIGN SPACE PARAMETER RANGES AND INCREMENTS FOR EXPERIMENTS ($\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}'_1$), AND TRAINING DATASET

Param	$\mathcal{S}_1$			$\mathcal{S}_2$			$\mathcal{S}'_1$			Training Dataset	
	$x_L - x_U$	dx	case/bits	$x_L - x_U$	dx	case/bits	$x_L - x_U$	dx	case/bits	$x_L - x_U$	dx
W_t	2-5	0.1	31/5	2-10	0.1	81/7	**2-10**	0.1	81/7	1-29	0.5
S_t	2-10	0.5	17/5	2-10	0.5	17/5	2-10	0.5	17/5	1-64	0.5
D_t	30-40	5	3/2	15-40	5	6/3	**15-40**	5	6/3	1-100	1
E_t	0-0.3	0.05	7/3	0-0.3	0.05	7/3	0-0.3	0.05	7/3	0-0.7	0.1
H_t	0.6-1.5	0.1	10/4	0.6-1.5	0.1	10/4	0.6-1.5	0.1	10/4	0.3-3.9	0.1
H_c	2-8	0.2	31/5	2-10	0.2	41/6	**2-10**	0.2	41/6	1-40	1
H_p	2-8	0.2	31/5	2-10	0.2	41/6	**2-10**	0.2	41/6	1-40	1
σ_t	3.8e+7-5.8e+7	1e+6	21/5	3.0e+7-5.8e+7	1e+6	29/5	3.8e+7-5.8e+7	1e+6	21/5	3.0e+7-5.8e+7	1e+6
R_t	-14.5-14	0.5	58/6	-14.5-14	0.5	58/6	-14.5-14	0.5	58/6	-14.5-14	0.5
Dk_t	2.5-4.5	0.05	41/6	2-5	0.05	61/6	2.5-4.5	0.05	41/6	1-7	0.1
Dk_c	2.5-4.5	0.05	41/6	2-5	0.05	61/6	2.5-4.5	0.05	41/6	1-7	0.1
Dk_p	2.5-4.5	0.05	41/6	2-5	0.05	61/6	2.5-4.5	0.05	41/6	1-7	0.1
Df_t	0.001-0.02	0.001	20/5	0.001-0.02	0.001	20/5	0.001-0.02	0.001	20/5	0.0001-0.1	0.0001
Df_c	0.001-0.02	0.001	20/5	0.001-0.02	0.001	20/5	0.001-0.02	0.001	20/5	0.0001-0.1	0.0001
Df_p	0.001-0.02	0.001	20/5	0.001-0.02	0.001	20/5	0.001-0.02	0.001	20/5	0.0001-0.1	0.0001
	7.14×10^{19} (2^{73})			2.97×10^{21} (2^{78})			6.53×10^{20} (2^{78})			1.31×10^{29}	

is considered as constraint. T3 and T4 incorporate *NEXT* to observe scenarios with more than two objectives. In addition, we conduct the experiment under the two different design search spaces $\mathcal{S}_1$ and $\mathcal{S}_2$ as illustrated in Table III. The last row presents the size of each search space. The design space $\mathcal{S}_1$ has solution space of 2^{73}, but contains the total number of 7.14×10^{19} valid cases. This discrepancy exists from generating invalid cases when mapping the possible cases to certain bits. During the optimization algorithm, we address this issue by evaluating the invalid cases and excluding from the performance evaluation. The design space $\mathcal{S}_2$ considers a larger space that includes $\mathcal{S}_1$.

To ensure the reliability of the approaches, we conduct repeat trials under the same conditions ten times. The final results are collected for each trial by running three EM simulations on the candidate selected by the ML surrogate model and HPO algorithm. It is important to note that the average runtime reported in the tables incorporates both the algorithm time and the EM simulation time. The average EM simulation time is 45.5 s for three EM simulation run in parallel. The success rate is determined by the number of trials where a solution satisfying the constraints is discovered. The better-final result is indicated by a lower f^{FoM}. Improvement of ISOP+ in the table represents the FoM improvement of ISOP+ over each method and is calculated as

$$f^{\text{FoM}} \text{ Impv. of ISOP+} = 100 \times \frac{f^{\text{FoM}}_{\text{method}} - f^{\text{FoM}}_{\text{ISOP+}}}{f^{\text{FoM}}_{\text{method}}}. \quad (12)$$

Therefore, any positive number presents a positive percentage improvement of ISOP+ over the other method.

Table IV compares average performance statistics for each method for T1 and T2. T1 and T2 are designed to evaluate the performance under the same design objectives with varying values, i.e., minimize L for varied Z_o and $Z_\pm$. We observe that ISOP+ achieves better efficiency and reliability in finding minimum points in all cases. For the SA approach, the success rate for T1/$\mathcal{S}_1$ did not reach 100% when a larger number of samples were observed compared to ISOP+ (SA-1). Additionally, despite the longer runtime and increased number of sample observed in SA-2, ISOP+ consistently outperformed SA by a maximum of 2.2% for all tasks. Compared to BO, ISOP+ achieves at most 30.8%, 33.6%, 32.7%, and 39.9% better performance for each task space. The results for $\mathcal{S}_2$ show that when the initial search space is larger, ISOP+ greatly outperforms compared to BO. This is because ISOP+ operates at a much faster pace, preventing slow BO from producing satisfactory results.

Table V presents the result comparison of each method on T3 and T4. T3 and T4 have the same constraint for Z as T1; however, they include *NEXT* in the metric as a constraint and FoM, respectively. T3 result shows that SA and BO both fail to find a viable solution as the search space becomes complex with *NEXT* involved as a constraint. ISOP+ demonstrates performance improvement of up to 19.1%, 55.2%, 5.8%, and 32.7% for each task and space, when given fewer resources compared to SA. Compared to BO, ISOP+ achieves better-FoM performance under similar runtime by 28.0%, 42.5%, 29.4%, and 40.6%, and also offers better-FoM performance with significantly less runtime. ISOP+ produced a lower-standard deviation for FoM metrics (L and *NEXT*), indicating that it is more reliable at finding good solution than other methods. The results for $\mathcal{S}_2$ indicate that ISOP+ performs better for all tasks when the initial search space is large. Furthermore, ISOP+ shows greater performance improvement in T3 and T4, compared to T1 and T2. This suggests that ISOP+ excels at finding good solutions as the solution space becomes more complicated with more objectives and constraints to consider.

B. Evaluation of ML Model Accuracy

We experiment with several basic regression models with cross-validation and including multiple iterations with HPO methods. The basic regression models we implemented includes DTR, GBR, PLR, RFR, SVR, XGBoost [28], and MLPR. As shown in Table III, we use the training dataset with a design space significantly larger than our target spaces, $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}'_1$. The 15 design parameters shown in the table are selected based on their impact on Z, L, and *NEXT*. Their ranges are decided based on the specification provided by

TABLE IV
EXPERIMENT RESULT COMPARISON FOR T1 AND T2 FOR SEARCH SPACE $\mathcal{S}_1$ AND $\mathcal{S}_2$

Task / $\mathcal{S}$	Method	Success. rate (succ./total)	Ave. run time (s)	Ave. sample seen	ΔZ		L		f^{FoM}	Impv. of ISOP+ (%)
					mean	stdev	mean	stdev		
T1/$\mathcal{S}_1$	SA-1	9/10	81.98	**16800**	0.379	0.101	-0.446	0.000	0.446	**2.2**
	SA-2	10/10	**88.11**	19981	0.384	0.073	-0.446	0.000	0.446	**2.2**
	BO-1	10/10	>1000	3039	0.459	0.079	-0.495	0.001	0.495	**11.8**
	BO-2	10/10	**86.64**	454	0.474	0.083	-0.630	0.010	0.630	**30.8**
	ISOP+	10/10	**73.04**	**16754**	0.570	0.030	-0.436	0.000	**0.436**	-
T1/$\mathcal{S}_2$	SA-1	10/10	82.67	**16750**	0.738	0.038	-0.303	0.000	0.303	**1.0**
	SA-2	10/10	**86.35**	18938	0.394	0.031	-0.302	0.000	0.302	**0.5**
	BO-1	10/10	>1000	2988	0.401	0.077	-0.343	0.001	0.343	**12.6**
	BO-2	10/10	**87.19**	425	0.540	0.096	-0.452	0.002	0.452	**33.6**
	ISOP+	10/10	**83.05**	**16675**	0.428	0.111	-0.300	0.000	**0.300**	-
T2/$\mathcal{S}_1$	SA-1	10/10	81.69	**16750**	0.311	0.024	-0.380	0.000	0.380	**1.5**
	SA-2	10/10	**87.45**	18538	0.256	0.016	-0.375	0.000	0.375	**0.2**
	BO-1	10/10	>1000	3049	0.544	0.202	-0.410	0.000	0.410	**8.8**
	BO-2	10/10	**85.96**	429	0.668	0.161	-0.556	0.011	0.556	**32.7**
	ISOP+	10/10	**77.73**	**16695**	0.441	0.135	-0.374	0.000	**0.374**	-
T2/$\mathcal{S}_2$	SA-1	10/10	83.07	**16700**	0.260	0.070	-0.262	0.000	0.262	**1.3**
	SA-2	10/10	**90.78**	20885	0.178	0.019	-0.260	0.000	0.260	**0.6**
	BO-1	10/10	>1000	2999	0.200	0.026	-0.321	0.001	0.321	**19.4**
	BO-2	10/10	**90.47**	458	1.001	0.319	-0.431	0.010	0.431	**39.9**
	ISOP+	10/10	**80.95**	**16691**	0.449	0.044	-0.259	0.000	**0.259**	-

TABLE V
EXPERIMENT RESULT COMPARISON FOR T3 AND T4 FOR SEARCH SPACE $\mathcal{S}_1$ AND $\mathcal{S}_2$

Task / $\mathcal{S}$	Method	Success rate (succ./total)	Ave. run time (s)	Ave. sample seen	ΔZ		L		$NEXT$		f^{FoM}	Impv. of ISOP+ (%)
					mean	stdev	mean	stdev	mean	stdev		
T3/$\mathcal{S}_1$	SA-1	4/10	83.26	**16800**	0.473	0.056	-0.489	0.002	-0.010	0.000	0.489	**7.8**
	SA-2	8/10	**82.83**	17973	0.410	0.070	-0.558	0.019	-0.010	0.000	0.558	**19.1**
	BO-1	6/10	>1000	3051	0.360	0.076	-0.518	0.003	-0.007	0.000	0.518	**12.9**
	BO-2	7/10	**85.35**	430	0.686	0.097	-0.627	0.007	-0.007	0.000	0.627	**28.0**
	ISOP+	10/10	**75.34**	**16754**	0.486	0.128	-0.451	0.000	-0.008	0.000	**0.451**	-
T3/$\mathcal{S}_2$	SA-1	8/10	82.99	**16750**	0.524	0.113	-0.691	0.286	-0.008	0.000	0.691	**55.2**
	SA-2	7/10	**89.13**	20325	0.274	0.079	-0.575	0.012	-0.013	0.000	0.575	**46.2**
	BO-1	4/10	>1000	2989	0.820	0.020	-0.355	0.000	-0.025	0.000	0.355	**12.9**
	BO-2	3/10	**92.44**	455	0.250	0.023	-0.538	0.019	-0.013	0.000	0.538	**42.5**
	ISOP+	10/10	**83.33**	**16716**	0.418	0.039	-0.309	0.000	-0.018	0.000 sz	**0.309**	-
T4/$\mathcal{S}_1$	SA-1	10/10	82.81	**16750**	0.270	0.018	-0.467	0.000	-0.006	0.000	0.479	**5.8**
	SA-2	10/10	**85.41**	18151	0.550	0.089	-0.460	0.000	-0.001	0.000	0.462	**2.4**
	BO-1	10/10	>1000	3052	0.378	0.126	-0.498	0.002	-0.002	0.000	0.502	**10.1**
	BO-2	10/10	**89.17**	429	0.448	0.108	-0.633	0.004	-0.003	0.000	0.639	**29.4**
	ISOP+	10/10	**74.38**	**16708**	0.312	0.067	-0.451	0.000	0.000	0.000	**0.451**	-
T4/$\mathcal{S}_2$	SA-1	10/10	82.75	**16650**	0.376	0.069	-0.473	0.000	-0.002	0.000	0.477	**32.7**
	SA-2	10/10	**99.61**	24709	0.407	0.059	-0.460	0.000	-0.002	0.000	0.464	**30.8**
	BO-1	10/10	>1000	2994	0.323	0.059	-0.386	0.003	-0.010	0.000	0.406	**20.9**
	BO-2	9/10	**99.71**	515	0.376	0.077	-0.490	0.004	-0.026	0.000	0.541	**40.6**
	ISOP+	10/10	**89.10**	**16600**	0.435	0.103	-0.305	0.000	-0.008	0.000	**0.321**	-

the Institute of Printed Circuits (IPCs) and the current manufacturing and fabrication capabilities. These 15 parameters satisfy our aim to include the parameters engineers control when designing a stack-up in a real-world industrial setting.

Table VI demonstrates that our 1D-CNN model outperformed other models in both MAE and MAPE for *Z* and *L*, and sMAPE for *NEXT*. We believe it is due to the complexity of functions between design parameters and performance

TABLE VI
EVALUATION OF TRAINED MODELS

ML Method	Z		L		$NEXT$	
	MAE	MAPE	MAE	MAPE	MAE	sMAPE
DTR	8.260	0.091	0.440	0.127	4.004	1.047
GBR	6.173	0.082	0.325	0.101	1.215	0.861
PLR	13.051	0.219	0.550	0.173	2.044	1.048
RFR	4.401	0.050	0.247	0.071	3.298	1.051
SVR	5.961	0.108	0.342	0.101	1.989	0.914
XGBoost	1.417	0.016	0.112	0.031	0.431	0.342
MLPR	0.459	0.006	0.053	0.016	0.203	0.442
1D-CNN [27]	**0.321**	**0.004**	**0.035**	**0.011**	**0.219**	**0.306**

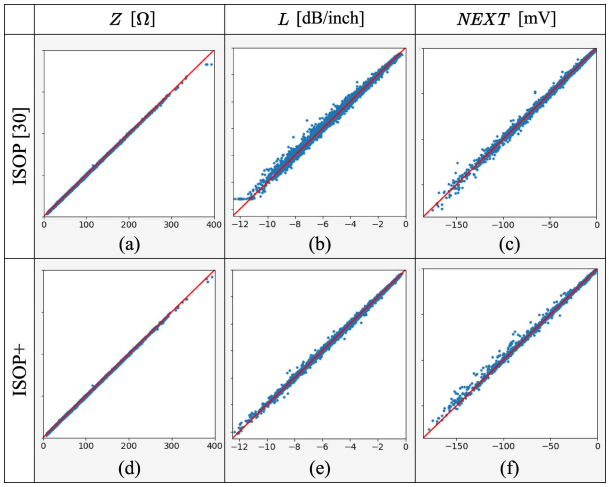


Fig. 6. Predicted performance versus ground truth. (a) MLP Z. (b) MLP L. (c) XGB *NEXT*. (d) 1D-CNN Z. (e) 1D-CNN L. (f) 1D-CNN *NEXT*.

metrics that CNN is capable of modeling. Our 1D-CNN model first extends the 15 features to 16 384 features using fully connected layer. Subsequently, we reshaped these features into a 2-D structure (2048×8). This process introduces a spatial and local correlation within the data. This approach proved beneficial, particularly when dealing with sparse training datasets, which accounted for only $7 \times 10^{-23}\%$ of the entire search space. By incorporating the spatial and local correlation, our ML model became better equipped to capture the intricate relationships between the features. Furthermore, in our implementation, we use leaky ReLU [32] as activation function and utilize dropout [33] to prevent over-fitting.

Fig. 6 illustrates the behavior of trained models for Z, L, and *NEXT*. The first row presents the result for MLP model for Z and L, and XGB model for *NEXT*, respectively, which are used in ISOP [30]. The second row presents 1D-CNN model for Z, L, and *NEXT*. The results demonstrate a strong correlation between the predicted performance values and the actual values, with the 1D-CNN model performing better than the other models. The high accuracy of the ML models allows us to accelerate the overall optimization flow by replacing the time-consuming simulations. In the other evaluations of the experiments, we choose the 1D-CNN model based on its lowest MAPE and sMAPE.

C. Case Study: Comparative Analysis

We compare our ISOP+ result with the previous study ISOP [30]. In our proposed method, we make improvements to both the ML surrogate model and the inverse optimization approach. Our ML surrogate model has improved by changing its architecture to 1D-CNN, and the inverse optimization method is enhanced with automatic tuning of the objective function and further optimization through gradient descent. The previous study's ML surrogate model is referred to as "MLP_XGB," while the new proposed model is referred to as "1D-CNN." We denote the previous study's inverse optimization method as "H," and the new method as "H_GD." For a fair comparison, we also implement another version where we use the improved ML surrogate model and the previous study's inverse optimization approach (H+1D-CNN). The scenario for "H_GD+MLP_XGB" cannot be evaluated due to the incompatibility of employing the gradient descent algorithm on the XGBoost model. The XGBoost model utilizes the gradient-boosted decision tree (GBDT) algorithm, which trains multiple decision trees on different subsets of the training dataset and subsequently combines them using gradients. As a result, the XGBoost model is not differentiable in the same way as neural networks, making it unsuitable for applying gradient descent to the model "MLP_XGB."

Tables VII and VIII present the detailed results of these comparisons. Figs. 7 and 8 present the summary of the results. The new proposed ISOP+ presents better FoM over all tasks and search spaces with less runtime and number of samples seen. Comparing "H+MLP_XGB" and "H+1D-CNN" reveals that the change made to ML surrogate model bring slight improvement compared to the previous study [30], but it also resulted in a longer runtime due to the computational complexity involved in evaluating a CNN model compared to MLP and XGBoost models. In conclusion, the combination of the improvement in the inverse optimization method and the ML surrogate model leads to a better performance in terms of FoM, along with a reduced runtime. Additionally, the new method eliminates the need for manual tuning of the objective function and the hyperparameters of the HPO method. This reduces the amount of manual effort required to run the framework and makes the result more reliable.

D. Case Study: Comparison With Manual Design and Adoption of Input Constraints

To acquire a comprehensive understanding of the result ISOP+ produces, we investigate one trial case and present the best candidates in that trial. ISOP+ is able to complete each task in a little over 1 min. As a comparison, we obtained a manual result from an experienced designer. The designer tries to optimize for loss when given the impedance target at 85 Ω with the acceptable tolerance of 1 Ω.

In the experiment, we utilize ISOP+ to solve each task in two different ways: 1) in the search space $\mathcal{S}_1$ without any input parameter constraints and 2) in the search space $\mathcal{S}'_1$ as illustrated in Table III with input constraints. $\mathcal{S}_1$ is manually defined with limited parameter ranges to prevent excessive horizontal width of the stack-up. On the other

TABLE VII
EXPERIMENT RESULT COMPARISON FOR DIFFERENT ISOP IMPLEMENTATIONS FOR T1 AND T2

Task / $\mathcal{S}$	Method		Ave. run time (s)	Ave. sample seen	ΔZ		L		f^{FoM}	Impv. of ISOP+ (%)
	optim technique	ML model			mean	stdev	mean	stdev		
T1/$\mathcal{S}_1$	H	MLP_XGB	82.95	25323	0.633	0.069	-0.445	0.000	0.445	2.0
	H	1D-CNN	85.09	25155	0.507	0.039	-0.449	0.000	0.449	2.7
	H_GD	1D-CNN	73.04	16754	0.570	0.030	-0.436	0.000	0.436	-
T1/$\mathcal{S}_2$	H	MLP_XGB	85.46	24971	0.374	0.118	-0.345	0.000	0.345	13.1
	H	1D-CNN	86.97	25822	0.444	0.039	-0.310	0.000	0.310	3.3
	H_GD	1D-CNN	83.05	16675	0.428	0.111	-0.300	0.000	0.300	-
T2/$\mathcal{S}_1$	H	MLP_XGB	88.74	25621	0.330	0.045	-0.451	0.000	0.451	17.1
	H	1D-CNN	90.30	26037	0.286	0.056	-0.393	0.000	0.393	4.8
	H_GD	1D-CNN	77.73	16695	0.441	0.137	-0.374	0.000	0.374	-
T2/$\mathcal{S}_2$	H	MLP_XGB	82.89	24407	0.908	0.376	-0.339	0.002	0.339	23.7
	H	1D-CNN	95.34	25439	0.780	0.377	-0.298	0.000	0.298	13.1
	H_GD	1D-CNN	80.95	16691	0.449	0.044	-0.259	0.000	0.259	-

TABLE VIII
EXPERIMENT RESULT COMPARISON FOR DIFFERENT ISOP IMPLEMENTATIONS FOR T3 AND T4

Task / $\mathcal{S}$	Method		Ave. run time (s)	Ave. sample seen	ΔZ		L		$NEXT$		f^{FoM}	Impv. of ISOP+ (%)
	optim technique	ML model			mean	stdev	mean	stdev	mean	stdev		
T3/$\mathcal{S}_1$	H	MLP_XGB	79.95	25781	0.537	0.102	-0.504	0.001	-0.013	0.001	0.504	1.6
	H	1D-CNN	86.17	24802	0.590	0.011	-0.509	0.002	-0.010	0.000	0.509	0.9
	H_GD	1D-CNN	75.34	16754	0.486	0.128	-0.451	0.000	-0.008	0.000	0.451	-
T3/$\mathcal{S}_2$	H	MLP_XGB	87.47	25123	0.517	0.069	-0.360	0.000	-0.035	0.000	0.360	14.0
	H	1D-CNN	91.60	25509	0.662	0.099	-0.349	0.000	-0.016	0.000	0.349	11.4
	H_GD	1D-CNN	83.33	16716	0.418	0.039	-0.309	0.000	-0.018	0.000	0.309	-
T4/$\mathcal{S}_1$	H	MLP_XGB	77.56	25797	0.584	0.012	-0.458	0.000	0.000	0.000	0.458	1.6
	H	1D-CNN	85.02	25761	0.474	0.090	-0.451	0.000	-0.002	0.000	0.455	0.9
	H_GD	1D-CNN	74.38	16708	0.312	0.067	-0.451	0.000	0.000	0.000	0.451	-
T4/$\mathcal{S}_2$	H	MLP_XGB	91.58	25294	0.560	0.157	-0.364	0.001	-0.006	0.000	0.376	14.6
	H	1D-CNN	102.71	25601	0.450	0.044	-0.356	0.000	-0.005	0.000	0.366	12.3
	H_GD	1D-CNN	89.10	16600	0.435	0.103	-0.305	0.000	-0.008	0.000	0.321	-

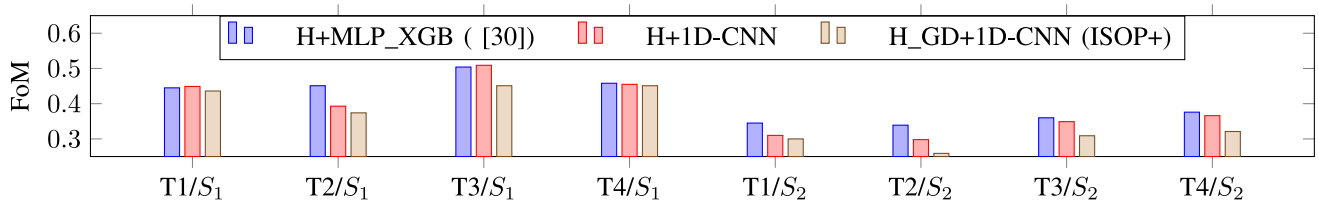


Fig. 7. Comparison of FoM with different optimization methods and ML surrogate models.

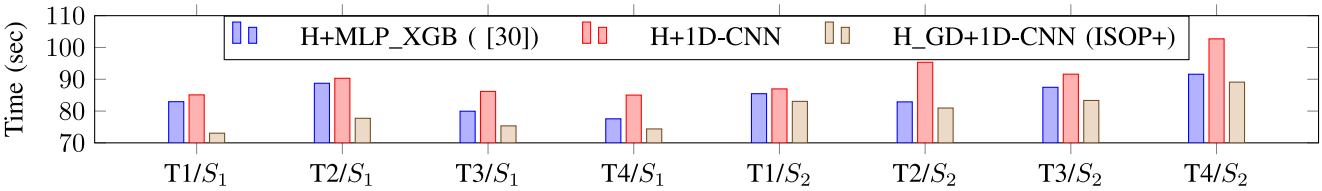


Fig. 8. Comparison of runtime with different optimization methods and ML surrogate models.

hand, $\mathcal{S}'_1$ encompasses $\mathcal{S}_1$ and has wider ranges for physical dimension parameters W_t, D_t, H_c, and H_p. By incorporating the input parameter constraints into $\mathcal{S}'_1$, we can ensure that the horizontal dimension does not become overly large. Consequently, the parameters W_t, S_t, D_t, H_c, and H_p become interdependent, and adjust to satisfy the constraints.

TABLE IX
EXPERIMENT RESULT COMPARISONS WITH MANUAL DESIGNS OVER DIFFERENT TASKS

Task	Method ($\mathcal{S}$ / IC?)	Design Parameters															Objectives			f^{FoM}
		W_t	S_t	D_t	E_t	H_t	σ_t	R_t	Dk_t	Df_t	H_c	Dk_c	Df_c	H_p	Dk_p	Df_p	Z	L	$NEXT$	
T1	Manual	5.0	6.0	20	0.00	1.5	5.8e+7	-14.5	4.30	0.001	8.0	4.30	0.001	8.0	4.30	0.001	85.69	-0.434	-2.77	0.434
T1	ISOP ($\mathcal{S}_1$/No)	5.0	6.5	30	0.00	1.5	5.8e+7	-14.5	4.50	0.001	6.2	4.50	0.001	8.0	3.55	0.001	85.70	-0.434	-0.49	0.434
	ISOP ($\mathcal{S}'_1$/Yes)	7.2	5.5	35	0.00	1.5	5.8e+7	-14.5	4.10	0.001	8.6	4.00	0.001	9.4	2.50	0.001	85.56	-0.361	-0.34	0.361
T3	ISOP ($\mathcal{S}_1$/No)	5.0	5.0	35	0.00	1.5	5.8e+7	-14.5	4.50	0.001	5.0	2.85	0.001	5.0	2.55	0.001	85.72	-0.439	-0.01	0.439
	ISOP ($\mathcal{S}'_1$/Yes)	8.2	3.5	40	0.30	0.7	5.7e+7	-14.5	2.50	0.001	8.0	2.80	0.001	8.0	3.35	0.001	84.94	-0.425	-0.04	0.425
T4	ISOP ($\mathcal{S}_1$/No)	5.0	6.0	40	0.00	1.5	5.8e+7	-14.5	2.50	0.001	4.6	4.50	0.001	6.4	2.50	0.001	85.74	-0.441	0.00	0.441
	ISOP ($\mathcal{S}'_1$/Yes)	7.9	4.0	35	0.30	1.5	5.8e+7	-14.5	2.50	0.001	7.0	2.50	0.001	7.0	2.55	0.001	85.07	-0.357	-0.06	0.477

The experiment conducts optimization tasks with three different input constraints:

1) $2 \times W_t + S_t \leq 20$,
   therefore, $f_1^{\text{IC}} = \max(2 \times W_t + S_t - 20, 0)$;
2) $D_t - 5 \times H_c \leq 0$,
   therefore, $f_2^{\text{IC}} = \max(D_t - 5 \times H_c, 0)$;
3) $D_t - 5 \times H_p \leq 0$,
   therefore, $f_3^{\text{IC}} = \max(D_t - 5 \times H_p, 0)$.

The constraints are established to limit the base width of a differential signal in a stripline structure. The first constraint is defined to control the base width of a differential signal pair, and the others determine the distance between differential pairs relative to the height of the insulating layers. An experienced designer in the industry defines these constraints to guarantee appropriate stack-up designs.

Table IX presents the result of manual design and the ISOP+ designs. The design parameter from T1 for $\mathcal{S}_1$ without the input constraints achieves the same L value with better *NEXT* than the manual, showing the ability of ISOP+ to find nonintuitive solutions to design expertise. Furthermore, T1 for $\mathcal{S}'_1$ with the input constraints produces better L and FoM compared to the manual design. Overall, the ISOP+ framework produces an excellent stack-up design that outperforms the manual. The result for T3 and T4 demonstrate that our framework can also perform multiobjective optimization to balance L and *NEXT*. By applying input constraints, we can guide the optimization tasks more efficiently. We believe that as the system becomes more dense and high-speed, the advantages of our flexible framework will continue to expand to enable global optimization over different performance metrics.

V. CONCLUSION

This article presents a novel framework, ISOP+, for automating stack-up design for advanced package design. The ISOP+ framework leverages an HPO search algorithm to identify the best-design parameters and optimize the performance of each stack-up layer. An ML-based surrogate model is used to accelerate the optimization process by replacing time-consuming simulations. Experimental results demonstrate that the ISOP+ framework can generate excellent design solutions in a matter of seconds. The proposed methodology offers an effective and efficient solution to automate the interconnect design for future packaging technology.

REFERENCES

[1] R. Mahajan, "Quiet revolutions: How advanced microelectronics packaging continues to drive heterogeneous integration," in *Proc. ITherm*, 2020, pp. 1408–1412.

[2] "Substrate-like PCB—The highlight in the future PCB industry." 2022. [Online]. Available: https://www.quick-pcba.com/pcb-news/substrate-like-pcb-technology.html

[3] M.-K. Shih, Y.-W. Huang, and G.-S. Lin, "Next-generation high-density PCB development by fan-out RDL technology," *IEEE Trans. Device Mater. Rel.*, vol. 22, no. 2, pp. 296–305, Jun. 2022.

[4] J. Cong, Z. Pan, L. He, C.-K. Koh, and K.-Y. Khoo, "Interconnect design for deep submicron ICs," in *Proc. ICCAD*, 1997, pp. 478–485.

[5] C.-L. Liao, B. Mutnury, C.-H. Chen, and Y.-J. Lee, "PCB stack-up design and optimization for next generation speeds," in *Proc. EPEPS*, 2016, pp. 155–158.

[6] J. He et al., "An integer programming application for PCB stack-up arrangement," in *Proc. EMC+SIPI*, 2019, pp. 432–436.

[7] Z. Kiguradze, J. He, B. Mutnury, A. Chada, and J. Drewniak, "Bayesian optimization for stack-up design," in *Proc. EMC+SIPI*, 2019, pp. 629–634.

[8] W. Jiang, K. Cai, B. Sen, and G. Wang, "Practical high speed PCB Stackup tool—Generation and validation," in *Proc. ECTC*, 2018, pp. 2288–2294.

[9] "Integrated channel analysis tool (ICAT)." 2023. [Online]. Available: https://designintools.intel.com/integrated-channel-analysis-tool-icat.html

[10] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020.

[11] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. NIPS*, 2012, pp. 1–9.

[12] J. Jung, A. B. Kahng, S. Kim, and R. Varadarajan, "METRICS2.1 and flow tuning in the IEEE CEDA robust design flow and OpenROAD," in *Proc. ICCAD*, 2021, pp. 1–9.

[13] M. M. Ziegler, J. Kwon, H.-Y. Liu, and L. P. Carloni, "Online and offline machine learning for industrial design flow tuning: (Invited—ICCAD Special Session Paper)," in *Proc. ICCAD*, 2021, pp. 1–9.

[14] Z. Xie et al., "FIST: A feature-importance sampling and tree-based method for automatic design flow parameter tuning," in *Proc. ASPDAC*, 2020, pp. 19–25.

[15] R. Liang et al., "FlowTuner: A multi-stage EDA flow tuner exploiting parameter knowledge transfer," in *Proc. ICCAD*, 2021, pp. 1–9.

[16] H. Geng, T. Chen, Y. Ma, B. Zhu, and B. Yu, "PTPT: Physical design tool parameter tuning via multi-objective Bayesian optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 1, pp. 178–189, Jan. 2023.

[17] C. Xu, G. Liu, R. Zhao, S. Yang, G. Luo, and Z. Zhang, "A parallel bandit-based approach for autotuning FPGA compilation," in *Proc. FPGA*, 2017, pp. 157–166.

[18] G. Murali, S. M. Shaji, A. Agnesina, G. Luo, and S. K. Lim, "ART-3D: Analytical 3D placement with reinforced parameter tuning for monolithic 3D ICs," in *Proc. ISPD*, 2022, pp. 97–104.

[19] A. Agnesina, K. Chang, and S. K. Lim, "VLSI placement parameter optimization using deep reinforcement learning," in *Proc. ICCAD*, 2020, pp. 1–9.

[20] B. Liu et al., "Analog circuit optimization system based on hybrid evolutionary algorithms," *Integration*, vol. 42, no. 2, pp. 137–148, 2009.

[21] W. Lyu et al., "An efficient Bayesian optimization approach for automated optimization of analog circuits," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 6, pp. 1954–1967, Jun. 2018.

[22] H. Wang et al., "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proc. DAC*, 2020, pp. 1–6.

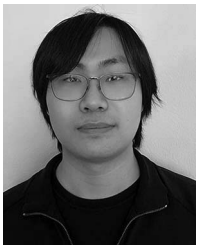
[23] E. Hazan, A. Klivans, and Y. Yuan, "Hyperparameter optimization: A spectral approach," in *Proc. ICLR*, 2018, pp. 1–18.

[24] P. Stobbe and A. Krause, "Learning fourier sparse set functions," in *Proc. AISTATS*, 2012, pp. 1–9.

[25] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6765–6816, 2017.
[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015, pp. 1–15.
[27] "Mechanisms of action (MoA) prediction: 2nd place solution—With 1D-CNN." 2020. [Online]. Available: https://www.kaggle.com/competitions/lish-moa/discussion/202256
[28] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. Data Min. Knowl. Disc.*, 2016, pp. 785–794.
[29] E. Bisong, *More Supervised Machine Learning Techniques with Scikit-Learn*. Berkeley, CA, USA: Apress, 2019, pp. 287–308.
[30] H. Chae et al., "ISOP: Machine learning-assisted inverse stack-up optimization for advanced package design," in *Proc. DATE*, 2023, pp. 1–6.
[31] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proc. Data Min. Knowl. Disc.*, 2019, pp. 2623–2631.
[32] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013, pp. 1–6.
[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.

Hyunsu Chae received the B.S. degree in semiconductor systems engineering from Sungkyunkwan University, Seoul, South Korea, in 2018. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA.

Her current research interests includes inverse design optimization and machine learning for package interconnect design.



Keren Zhu (Member, IEEE) received the B.S. degree in electrical engineering from the University of Wisconsin–Madison, Madison, WI, USA, in 2016, and the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2022.

He is a Postdoctoral Research Fellow with the Department of Electrical and Computer Engineering, The University of Texas at Austin. His research interests include physical design automation, analog integrated circuit design automation, machine learning for EDA, and computing system with emerging technologies.

Bhyrav Mutnury (Fellow, IEEE) received the M.Sc. and Ph.D. degrees from the Georgia Institute of Technology, Atlanta, GA, USA, in 2002 and 2005, respectively.

He is a Senior Distinguished Engineer and SI Team Leader with ISG Group, Dell Technologies, Austin, TX, USA, where he is responsible for storage, network, rack, and blade server designs. He has authored and coauthored more than 90 refereed publications in various IEEE and non-IEEE conferences and has more than 200 patents issued.

Douglas E. Wallace Jr. received the B.S. degree from The University of Texas at Austin, Austin, TX, USA, in 1981.

He is a Senior Electrical Principal Engineer with Dell Technologies, Austin. His work and interests with Dell has been in signal integrity high-speed signal simulation automation. His previous experience has been in military/medical fields designing filtering, control software, and hardware for embedded-processor systems.

Douglas S. Winterberg received B.S. degree from Rochester Institute of Technology, Rochester, NY, USA, in 1996.

He is a Technical Staff Engineer with Dell Technologies, Austin, TX, USA. His work and interests with Dell has been in signal integrity high-speed signal simulation and validation. His previous experience has been in computer system design and validation.

Daniel de Araujo (Senior Member, IEEE) received the B.S. degree in electrical engineering from Michigan State University, East Lansing, MI, USA, in 1997, and the first M.S. degree in computer engineering from North Carolina State University, Raleigh, NC, USA, in 2000. He is currently pursuing the second M.S. degree in computer science with the University of Texas at Austin, Austin, TX, USA.

He has 26 years of experience in board, package, and chip design, simulation, and validation. He has worked with IBM, Zürich, Switzerland; Ansoft, Austin; and Physware/Nimbic/Mentor Graphics/Siemens, Cedar Park, TX, USA, where he is a Principal Product Architect for the HyperLynx Advanced Solvers. He has 14 patents issued, 11 filed, nine patent disclosure publications, and 81 peer-reviewed publications in international IEEE conference proceedings, transactions, journals, and books.

Jay Reddy is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Texas at Austin, Austin, TX, USA.

He is a Director with Dell Technologies, Austin, TX, USA. He is a part of the leadership team that made Dell servers the number one choice in corporate data centers worldwide. He is a part of the team that won an NSF Grant to create the Institute for Foundations of Machine Learning, University of Texas at Austin.

Adam Klivans received the B.S. degree and the M.S. degree in 1997 from Carnegie-Mellon University, Pittsburgh, PA, USA, and the Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2002.

He is a Professor of Computer Science with the University of Texas at Austin, Austin, TX, USA, and the Director of the NSF AI Institute for Foundations of Machine Learning.

David Z. Pan (Fellow, IEEE) received the B.S. degree from Peking University, Beijing, China, in 1992, and the M.S. and Ph.D. degrees from The University of California at Los Angeles, Los Angeles, CA, USA, in 1998 and 2000, respectively.

He is currently a Professor and Silicon Labs Endowed Chair with the University of Texas at Austin, Austin, TX, USA. His research interests include electronic design automation, synergistic AI and IC co-optimizations, and DFM. He has published over 450 peer-reviewed papers and holds eight U.S. patents. He has served in many committees, such as DAC 2023 Program Co-Chair.

Dr. Pan is a Fellow of ACM and SPIE.