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

Hyunsu Chae[1], Bhyrav Mutnury[3], Keren Zhu[1], Douglas Wallace[3], Douglas Winterberg[3], Daniel de Araujo[4], Jay Reddy[3], Adam Klivans[2] and David Z. Pan[1]
ECE Department, The University of Texas at Austin, Austin, TX, USA
CS Department, The University of Texas at Austin, Austin, TX, USA
Dell Infrastructure Solutions Group, Round Rock, TX, USA
Electronic Board Systems, Siemens EDA, Austin, TX, USA
hyunsu.chae@utexas.edu, jay.reddy@dell.com, dpan@ece.utexas.edu

***Abstract*—Future computing calls for heterogeneous integration, e.g., the recent adoption of the chiplet methodology. However, high-speed cross-chip interconnects and packaging shall be 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 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 optimizations. However, stack-up design is usually done manually in the industry, where experienced designers may devote many hours to adjusting the physical dimensions and materials to meet the desired specifications. This process, however, is time-consuming, tedious, and sub-optimal, largely depending on the designer's expertise. In this paper, we propose to automate the stack-up design with a new framework, *ISOP*, using machine learning for inverse stack-up optimization for advanced package design. Given a target design specification, ISOP automatically searches for ideal stack-up design parameters while optimizing performance. We develop a novel machine learning-assisted hyper-parameter optimization method 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.**

I. Introduction

Advances in packaging technologies are driving the scaling of electronics. Amid the slowdown of the integrated circuits (IC) 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 rapidly evolving. 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. However, the signals on such PCBs are sensitive to the physical stack-up design. In a typical industry setting, the stack-up design is conducted manually by experienced engineers through many trial-and-error iterations to ensure signal integrity.

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. 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 reasonable 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] proposed using the Bayesian optimization (BO) method to optimize the five-parameter stack-up design, however, fail to provide a fully automated and scalable solution to the stack-up designs.

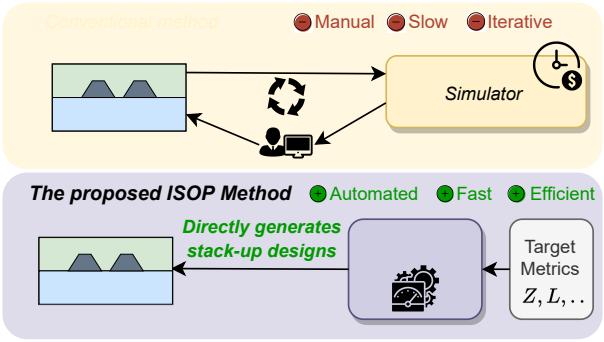


Fig. 1: An illustration of the conventional stack-up design and our proposed Inverse Stack-up Optimization Framework (ISOP).

In this paper, we propose a fully automated stack-up design methodology in an inverse optimization setting. Instead of assisting the manual design, our proposed framework, inverse stack-up optimization for advanced package design (ISOP), directly produces the stack-up designs 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 L at various frequencies and cross-talk. The main contributions of this work are summarized as follows.

- We propose an agile HDI PCB stack-up design framework, ISOP, which automates the stack-up design process and outperforms manual design. 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.
- We formulate the inverse stack-up optimization as a hyper-parameter optimization (HPO) problem. An effective two-stage HPO search algorithm is developed to solve the stack-up design

efficiently.

- We accelerate the HPO 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.
- Experimental results demonstrate that the ISOP framework reduces the design cycle from hours to minutes and outperforms other baseline optimization methods.

The rest of this paper is organized as follows. We introduce the stack-up design problem and its formulation in Section II. Section III describes the details of our proposed framework and its major components. Section IV presents the experimental results with other optimization methods. 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

A PCB's construction begins with designing its material layers' arrangement, i.e., stack-up. PCB consists of mainly passive components, and its primary function is to transfer a signal from one port to another while maintaining signal integrity. A layer's physical dimensions and material properties determine the transmission line performance. A simplified structure of a single differential strip-line layer is illustrated in Fig. 2. The subscript t, c, and p denote a layer's metal trace, glass-reinforced epoxy laminate sheet (core), and pre-impregnated bonding sheet (pre-preg), respectively. Parameters H, W, S, D, and E represent the height, width, the spacing between differential signals, the distance between two differential pairs, and an etch factor representing the trapezoidal shape of the metal trace, respectively. Material properties Dk, Df, R, and C denote dissipation factor, dielectric constant, surface roughness, and conductivity, respectively.

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]. Software based on integrated channel analysis tool (ICAT) [9] is one example of an EM simulation tool. It takes about two minutes to compute performance metrics for each stack-up design. 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 crosstalk. 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 non-intuitive solutions. Our studies show that manual stack-up designs often result in inferior quality, especially when facing trade-offs 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.

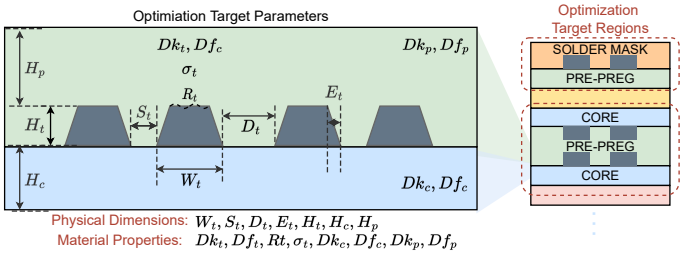


Fig. 2: Structure of a Differential Stripline Layer

B. Hyper-parameter optimization

Hyper-parameter optimization (HPO) searches for the best set of parameters in an optimization problem that is costly to evaluate in general. An HPO aims to obtain a parameter set $\mathbf{x}$ as shown in (1).

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

where $\mathbf{x}$ denotes the hyper-parameters and $f(\mathbf{x})$ is the objective function to be minimized. Unlike traditional optimization problems, the objective functions in HPO are usually non-convex and non-differentiable, 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 are 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, use a heuristic algorithm to guide the randomized search process to improve efficiency. In recent years, Bayesian optimization (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. It uses an acquisition function to balance exploration and exploitation 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]–[16] and Field-Programmable Gate Array (FPGA) [17]. It can also be used to optimize the hyper-parameters 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 hyper-parameters and applies HPO to find the sizing solution. There have been a variety of HPO search algorithms applied to the analog sizing problem, including 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 (FoM). In science and mathematics, an inverse problem often refers to estimating the unknown parameters inversely through measurements. Similarly, our optimization scheme searches for 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 (2).

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

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, and f^C denotes the performance specifications constraints. k is the number of constraints.

III. Algorithms

This section presents our ISOP framework and inverse stack-up design optimization. It shall be noted that our framework can be easily extended to other advanced packaging designs where stack-up design and optimizations are needed.

A. Overall flow

The inverse stack-up design optimization aims to find the optimal set of design parameters for each layer of a PCB's stack-up. The final stack-up design must meet performance specifications and optimize a specified performance FoM objective function. Both the constraints and FoM are from performance metrics and are non-trivial to evaluate. Traditional manual design flow rely on an engineer's experience and trial-and-error approach using slow simulations. ISOP offers a more efficient and automated alternative.

The ISOP framework solves the inverse PCB stack-up optimization by incorporating a discrete domain HPO. Fig. 3 shows an illustration of the overall flow. It takes an FoM function, a set of performance constraints, and a set of parameter search spaces as inputs and the stack-up design parameters as output. The HPO process contains two stages: early search exploration and later candidate roll-out. The first stage samples the parameters globally to explore the search space. The second stage then chooses the final stack-up designs based on the results from the first stage.

Algorithm 1 shows the details procedures of our ISOP framework. We first encode an initial search space based on the input parameter search range (line 1). In the exploration stage, the search space is iteratively shrunk to prune the non-ideal stack-up parameters based on an optimization objective function $\hat{g}(\cdot)$ (lines 3-6). Instead of using time-consuming EM simulations, we sample the performance metrics from ML surrogate model. We intend to obtain more samples and rapidly reduce the search space in a trade-off of some accuracy. After exploring the search space, we roll out $cand\_num$ design candidates from the reduced search space. (lines 7-10) We further evaluate them with accurate EM simulations and choose the final solution based on an objective function $g(\cdot)$. ISOP can generate multiple design candidates ranked by FoM in the roll-out stage if specified by the user.

In the rest of the section, the details of the ISOP framework algorithm are presented.

B. HPO Search Algorithm

ISOP adopts and adapts the Harmonica algorithm [23] in search space exploration. Harmonica is a spectral approach to discrete domain HPO. The core principle is that an objective can be modeled as a sum of sparse and low-degree Fourier polynomials. Harmonica invokes the polynomial sparse recovery (PSR) subroutine to reduce the search space iteratively. Equation (3) shows the simplified version of Harmonica's PSR subroutine.

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

where $c_1, \cdots, c_k$ is the indices of the most significant k components of the Fourier coefficients α, and $\mathcal{F}(\cdot)$ denotes the objective function, which in our case is $\hat{g}(\cdot)$. This approximation method can efficiently reconstruct the objective function with limited randomized samples [24]. In the search process, the Harmonica algorithm iteratively fixes the significant variables based on the $p(\mathbf{x})$ polynomial approximation to reduce the search space. Each iteration selects a set of variable combinations from the reduced search space and further evaluates with the performance models. In our implementation, the variable combinations are chosen based on the Hyperband [25] algorithm, which is a bandit algorithm that balances the exploration versus exploiting, given a limited sampling budget. In our experiments, the Hyperband greatly outperforms the naive random sampling method.

The proposed HPO search algorithm allows parallelized design parameter sampling in the ISOP framework. In each iteration, the ISOP framework evaluates multiple candidate parameter sets in parallel. The batched samples can be combined into the polynomial approximation to fit the optimization objective function. Compared to the conventional sequential HPO algorithms, such as Bayesian optimization, the proposed parallelized method allows us to obtain more samples in the same runtime budget and produce better optimization results.

The Harmonica algorithm can in-situ optimize over a constrained and discrete parameter search space. Due to material and technology selection limitations, the valid stack-up parameters are limited to discrete values and specific ranges. The conventional HPO algorithm assumes a continuous space, and this gap can degrade the HPO results. In ISOP, we encode the entire discrete search space $\mathcal{S}$ to a binary search space $T(\mathbf{x})$. The variable combination to be evaluated is directly chosen from the encoded space, ensuring that it is within the defined search space $\mathcal{S}$. Such property benefits the overall efficiency of the optimization process by avoiding explorations with invalid design parameter values.

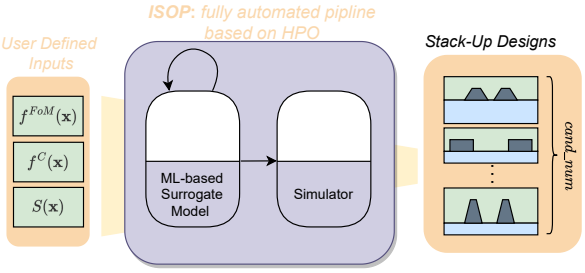


Fig. 3: Overall Flow of ISOP framework.

Algorithm 1 ISOP

Input: $f^{FoM}(\cdot)$, $f^{C}(\cdot)$, $\mathcal{S}$
Output: $cand\_num$ combination of $\mathbf{x}^*$

1: `Encode` $\mathbf{x}$ and $\mathcal{S}$
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: $T(\mathbf{x}) \leftarrow$ `update_space`$(T(\mathbf{x}),\ \mathbf{x}^q,\ \hat{g}(\hat{M}(\cdot)))$
6: **end for**
7: **for** $i \leftarrow 1$ to $cand\_num$ **do**
8: $\mathbf{x}_i \leftarrow T(\mathbf{x})$
9: $\mathbf{y}_i \leftarrow g(M(\mathbf{x}_i))$
10: **end for**

C. ML-based Surrogate Models

We use an ML-based surrogate model to replace the expensive EM simulation in the early exploration to accelerate the optimization process. Our HPO procedures require frequent performance evaluations on different design parameters. Relying on the time-consuming EM simulations slows down the process and limits the number of samples we can afford. Therefore, we use an ML-based surrogate model to increase the number of samples observed by the HPO search algorithm within the runtime budget.

Building our ML surrogate models is essentially a regression problem with tabular features. We construct one regression model explicitly tailored for each performance metric. Our dataset contains 90k unique stack-up design combinations, of which 80% are used for training and 20% for testing. Using an industry-standard simulation tool based on ICAT [9], we randomly query the data throughout the wide range of each parameter. The design parameters and their ranges represent a large solution space, 10^{29}, and are set by designers. Preprocessing of the dataset includes parameter normalization and feature engineering to add relevant features based on transmission physics understanding. For instance, a feature is included to show when the core height or pre-preg height is particularly small, indicating when the electric field between the copper planes becomes extremely high and causes the transmission line to have high capacitance. While the training dataset is equivalent to only $7 \times 10^{-23}\%$ of the entire search space, our ML surrogate models empirically result in satisfying accuracy and effectively guide the search space exploration.

There are a variety of regression methods, including decision tree (DT), gradient boosting regressor (GBR), random forest (RF), support vector machine (SVM), multi-layer perceptron (MLP), and XGBoost [26]. In this work, we empirically choose the MLP model for impedance and loss and XGBoost for cross-talk mode based on the test accuracy. 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 for impedance and loss, and symmetric mean absolute percentage error (sMAPE) for cross-talk as it could have zero values. The comparison study of the different models and their accuracy is presented in Section IV-B.

D. Optimization Objective Function

We introduce an objective function $\hat{g}(\cdot)$ for optimization in the early search space exploration stage. The inverse stack-up optimization problem is to minimize f^{FoM} while honoring the constraint f^C. However, directly solving the constraint problem is difficult and inefficient. Evaluation of f^{FoM} and f^C requires querying from performance models. 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 (4).

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{X}} g(\mathbf{x}) \tag{4}$$

$$g(\mathbf{x}) = \sum_i w_i^{FoM} \cdot f_i^{FoM}(\mathbf{x}) + \sum_j w_j^C \cdot f_j^C(\mathbf{x}), \quad \text{where } f_j^C(\mathbf{x}) = \max(M_j(\mathbf{x}) - f_{j\pm}, 0). \tag{5}$$

The f_j^C becomes a clip function for constraints. For instance, one may wish to give a constraint for impedance such that it has an acceptable tolerance, $Z_\pm$, of the characteristic impedance Z_o, in which $f_Z^C(\mathbf{x}) = \max(|M_Z(\mathbf{x}) - Z_o| - Z_\pm, 0)$.

We further smooth $g(\cdot)$ into our optimization 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 non-differentiable 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 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.

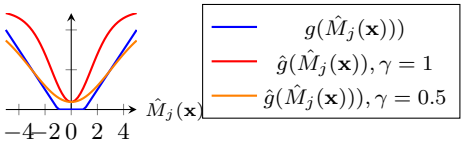


Fig. 4: Function $g(\cdot)$ and $\hat{g}(\cdot)$ with different γ.

$$\hat{g}(\mathbf{x}) = \sum_i w_i^{FoM} \cdot f_i^{FoM}(\mathbf{x}) + \sum_j w_j^C \cdot \hat{f}_j^C(\mathbf{x}), \quad \text{where} \\ \hat{f}_j^C(\mathbf{x}) = (S(\gamma \cdot \hat{M}_j(\mathbf{x}) - f_{j\pm}) + S(-\gamma \cdot \hat{M}_j(\mathbf{x}) - f_{j\pm})) \tag{6}$$

Our framework utilizes both $g(\cdot)$ and $\hat{g}(\cdot)$. Fig. 4 gives a comparison between the two. We can adjust $\hat{g}(\cdot)$ furthermore for our objective 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. We empirically choose γ and weights w^{FoM} and w^C in the experiments.

IV. Experimental Results

In this section, we present experimental results for the ISOP framework to prove its effectiveness and reliability. We implement all the algorithms in Python and report performance metrics verified with EM simulations in an industrial setting. We first evaluate the effectiveness of the proposed inverse optimization method in Section IV-A. The accuracy evaluation of our ML-based surrogate model is in Section IV-B. The Section IV-C presents a comparative study between the ISOP-generated and the manual stack-up design.

A. Evaluation of the HPO framework

This section describes the experimental results for the proposed framework and baseline approaches. No current research, to our knowledge, serves as a baseline for this problem. Therefore, utilizing the same surrogate model, we compare our method to two other well-known optimization techniques, simulated annealing (SA) and Bayesian optimization (BO). We further adjust the hyper-parameters of SA and BO to match either the runtime (denoted as "-1") or the number of observed samples (denoted as "-2") of the proposed ISOP framework. For example, SA-1 is the SA algorithm with similar runtime to ISOP, and SA-2 is when similar number of samples are observed as ISOP. The optimization is terminated after 1500 seconds. The BO algorithm is slow due to its sequential process, and BO-2 experiments are terminated earlier. In addition, the experiment conducts optimization tasks with 4 different user objectives and constraints as shown in Table I. Z is differential impedance, $|Z_o|$ is the transmission line's characteristic differential impedance, $Z_\pm$ is

TABLE I: Description of Each Experiment Tasks

Tasks	f^{FoM}	f^C	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	-	-

TABLE II: Design Space Parameter Ranges and Increments for Experiments ($\mathcal{S}$)

W_t	2-5, 0.1	S_t	2-10, 0.5	D_t	30-40, 5
E_t	0-0.3, 0.05	H_t	0.6-1.5, 0.1	C_t	3.8e+7-5.8e+7, 1e+6
R_t	-14.5-14, 0.5	Dk_t	2.5-4.5, 0.05	Df_t	0.001-0.02, 0.001
H_c	2-8, 0.2	Dk_c	2.5-4.5, 0.05	Df_c	0.001-0.02, 0.001
H_p	2-8, 0.2	Dk_p	2.5-4.5, 0.05	Df_p	0.001-0.02, 0.001

TABLE III: Experiment Result Comparison for T1 and T2

Tasks	Methods	Success rate (success/total)	Ave. run time (s)	Ave. sample seen	ΔZ		L		f^{FoM}	f^{FoM} Impv. of ISOP (%)
					mean	stdev	mean	stdev		
T1	SA-1	10/10	535	100,000	0.590	0.258	-0.451	0.010	0.4512	**2.41**
	SA-2	10/10	402	60,000	0.349	0.243	-0.458	0.011	0.4576	**3.78**
	BO-1	10/10	411	1,908	0.443	0.346	-0.498	0.037	0.4980	**11.58**
	BO-2	10/10	>1500	5,155	0.483	0.145	-0.473	0.006	0.4733	**6.78**
	ISOP	10/10	395	62,260	0.534	0.313	**-0.440**	**0.005**	**0.4403**	-
T2	SA-1	10/10	525	70,000	0.055	0.021	-0.473	0.015	0.4725	**2.54**
	SA-2	10/10	504	64,000	0.370	0.331	-0.469	0.040	0.4693	**1.86**
	BO-1	10/10	503	1,800	1.232	0.374	-0.742	0.279	0.7424	**37.97**
	BO-2	10/10	>1500	4,799	0.895	0.533	-0.585	0.135	0.5852	**21.30**
	ISOP	10/10	483	63,290	0.480	0.461	**-0.461**	**0.009**	**0.4605**	-

TABLE IV: Experiment Result Comparison for T3 and T4

Tasks	Methods	Success rate (success/total)	Ave. run time (s)	Ave. sample seen	ΔZ		L		$NEXT$		f^{FoM}	f^{FoM} Impv. of ISOP (%)
					mean	stdev	mean	stdev	mean	stdev		
T3	SA-1	1/10	375	22,000	0.010	N /A	-2.054	N /A	0.000	N /A	2.0540	**77.49**
	SA-2	2/10	666	55,000	0.445	0.120	-1.107	0.728	-0.020	0.028	1.1065	**58.22**
	BO-1	10/10	390	1,310	0.484	0.324	-0.585	0.141	-0.027	0.016	0.5852	**21.00**
	BO-2	10/10	>1500	3,822	0.552	0.354	-0.544	0.032	-0.010	0.012	0.5436	**14.95**
	ISOP	10/10	364	54,944	0.650	0.269	**-0.462**	**0.008**	-0.021	0.020	**0.4623**	-
T4	SA-1	1/10	405	23,000	0.080	N /A	-0.993	N /A	0.000	N /A	0.9930	**54.66**
	SA-2	1/10	573	46,000	0.080	N /A	-1.067	N /A	0.000	N /A	1.0670	**57.81**
	BO-1	10/10	410	1,400	0.577	0.390	-0.594	0.103	-0.003	0.007	0.6000	**24.97**
	BO-2	10/10	>1500	3,927	0.470	0.298	-0.515	0.025	-0.002	0.004	0.5186	**13.20**
	ISOP	10/10	411	46,000	0.628	0.176	**-0.450**	**0.003**	**0.000**	**0.000**	**0.4502**	-

TABLE V: Design Space Parameter Ranges and Increments for Training Dataset

W_t	1-29, 0.5	S_t	1-64, 0.5	D_t	1-100, 1
E_t	0-0.7, 0.1	H_t	0.3-3.9, 0.1	C_t	3.0e+7-5.8e+7, 1e+6
R_t	-14.5-14, 0.5	Dk_t	1-7, 0.1	Df_t	0.0001-0.1, 0.0001
H_c	1-40, 1	Dk_c	1-7, 0.1	Df_c	0.0001-0.1, 0.0001
H_p	1-40, 1	Dk_p	1-7, 0.1	Df_p	0.0001-0.1, 0.0001

the acceptable tolerance of $|Z - Z_o|$, L is differential insertion loss at 16GHz in dB/inch, and $NEXT$ is the peak near-end differential cross-talk. T3 and T4 incorporate $NEXT$ to observe scenarios with more than two objectives.

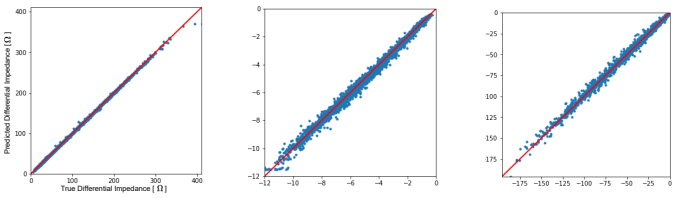
To ensure the reliability of the approaches, we conduct repeat trials under the same conditions ten times. The final results are collected for each trial by running ten EM simulations on the candidate selected by the ML surrogate model and HPO algorithm. The success rate is the number of times a solution satisfying the constraint is discovered. The design search space used in the experiments is described in Table II. It shows that the solution space is larger than 10^{19}.

Table III 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 target Z_o and tolerance $Z_{\pm}$. We observe that ISOP achieves better efficiency and reliability in finding minimum points in all task cases. f^{FoM} improvement of ISOP is calculated by $100 \times (f^{FoM}_{method} - f^{FoM}_{ISOP})/f^{FoM}_{method}$, and ISOP achieves at most 2.54%, 3.78%, 37.97%, and 21.30% better performance compared to SA-1, SA-2, BO-1, and BO-2, respectively. ISOP produced a lower standard deviation for L, indicating that it is more reliable than other methods.

Table IV 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. The result shows that SA fails to find a viable solution as the search space becomes complex with $NEXT$ involved in the metric. BO can also find feasible results, but ISOP finds better answer consistently at all times. ISOP achieves better FoM performance under similar runtime compared to BO by 21.00% and 24.97% for T3 and T4. Also, ISOP offers better FoM performance with significantly less runtime.

TABLE VI: Evaluation of Trained Models

ML method	Z		L		$NEXT$	
	MAE	MAPE	MAE	MAPE	MAE	sMAPE
DT	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
PL	13.051	0.219	0.550	0.173	2.044	1.048
RF	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
XGB	1.417	0.016	0.112	0.031	0.431	**0.342**
MLP	**0.459**	**0.006**	**0.053**	**0.016**	**0.203**	0.442



(a) (b) (c)

Fig. 5: Predicted performance versus ground truth. (a) Z. (b) L. (c) $NEXT$.

B. Evaluation of ML Model Accuracy

We experiment with several basic regression models with cross-validation, and including multiple iterations with HPO methods. As shown in Table V, we use the training dataset with a design space significantly larger than our target space. Table VI demonstrates that MLP and XGBoost performed significantly better than others in both MAE and MAPE. Overall, MLP produces the best MAE for all three tasks We believe it is due to the complexity of functions between design parameters and performance metrics that MLP is capable of modeling. XGBoost also produces good accuracy compared to other methods. Fig. 5 illustrates the behavior of trained models for Z, L, and

TABLE VII: Experiment Result Comparisons with Manual Designs over Different Tasks

Methods	Tasks	top	Design Parameters (W_t, S_t, D_t, E_t, H_t, C_t, R_t , Dk_t, Df_t , H_c, Dk_c, Df_c, H_p, Dk_p, Df_p)	Objectives Z	L	$NEXT$
Manual	T1	1	(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
ISOP	T1	1	(5.0, 5.5, 30, 0.00, 1.5, 5.8e+7, -14.5, 3.70, 0.001, 7.2, 4.10, 0.001, 7.0, 3.70, 0.001)	85.91	-0.434	-0.26
		2	(5.0, 5.5, 30, 0.05, 1.5, 5.8e+7, -14.5, 3.45, 0.001, 5.8, 4.25, 0.001, 8.0, 3.55, 0.001)	85.31	-0.441	-0.19
		3	(5.0, 5.5, 40, 0.05, 1.5, 5.8e+7, -14.5, 2.80, 0.001, 5.6, 4.40, 0.001, 8.0, 3.65, 0.001)	84.98	-0.443	-0.02
ISOP	T3	1	(5.0, 9.0, 40, 0.00, 1.5, 5.8e+7, -14.5, 3.85, 0.001, 5.2, 3.15, 0.001, 4.0, 3.25, 0.001)	85.58	-0.457	0
		2	(4.9, 8.0, 30, 0.00, 1.5, 5.8e+7, -14.5, 4.00, 0.001, 6.2, 3.80, 0.001, 4.0, 3.20, 0.001)	84.82	-0.463	-0.04
		3	(5.0, 8.0, 40, 0.15, 1.5, 5.8e+7, -14.5, 2.70, 0.001, 5.4, 3.95, 0.001, 4.0, 3.20, 0.001)	84.99	-0.475	0
ISOP	T4	1	(5.0, 8.0, 40, 0.00, 1.5, 5.8e+7, -14.5, 4.00, 0.001, 4.8, 4.15, 0.001, 6.0, 3.15, 0.001)	85.69	-0.447	0
		2	(5.0, 7.5, 40, 0.10, 1.5, 5.8e+7, -14.5, 3.85, 0.001, 5.2, 4.50, 0.001, 5.0, 2.65, 0.001)	85.59	-0.451	0
		3	(4.9, 6.5, 40, 0.05, 1.5, 5.8e+7, -14.5, 3.95, 0.001, 5.6, 4.10, 0.001, 4.5, 2.75, 0.001)	84.76	-0.456	0

$NEXT$, and shows that the predicted performances are well correlated with the golden values. 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 models based on the lowest MAPE and sMAPE (MLP for Z/L and XGBoost for $NEXT$).

C. Case Study: Comparisons with Manual Designs

To acquire a comprehensive understanding of the time-efficient result ISOP produces, we investigate one trial case and present the three best candidates from that trial. 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.

Table VII presents the result of manual and ISOP design. Overall, the ISOP framework produces an excellent stack-up design comparable to the manual. The top-1 design parameter from ISOP for T1 achieves the same L as the manual. It demonstrates the capability of ISOP to find non-intuitive solutions to design expertise. Our framework can also perform multi-objective optimization to balance L and $NEXT$. 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 paper presents a novel framework, ISOP, for automating stack-up design for advanced package design. ISOP leverages a HPO search algorithm to find the design parameters and optimize for layer performance. 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 produce excellent design solutions in minutes. The proposed methodology provides an effective and efficient solution to automate the interconnect design for future packaging technology.

Acknowledgment

This work is supported in part by NSF under grants 1718570, 2019844, and 2112665, and by the NSF AI Institute for Foundations of Machine Learning (IFML).

References

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

[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 TDMR*, 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.

[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.

[6] J. He, A. Klivans, Z. Kiguradze, A. Chada, B. Mutnury, J. Reddy, J. Drewniak, and J. Fan, "An integer programming application for PCB stack-up arrangement," in *Proc. EMC+SIPI*, 2019.

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

[8] W. Jiang, k. Cai, B. Sen, and G. Wang, "Practical high speed pcb stackup tool - generation and validation," in *Proc. ECTC*, 2018.

[9] "Integrated channel analysis tool (ICAT)." [Online]. Available: https://designintools.intel.com/Integrated_Channel_Analysis_Tool_ICAT_p/stlgrn160.htm

[10] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, 2020.

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

[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.

[13] M. M. Ziegler, J. Kwon, H.-Y. Liu, and L. P. Carloni, "Online and offline machine learning for industrial design flow tuning," in *Proc. ICCAD*, 2021.

[14] Z. Xie, G.-Q. Fang, Y.-H. Huang, H. Ren, Y. Zhang, B. Khailany, S.-Y. Fang, J. Hu, Y. Chen, and E. C. Barboza, "FIST: A feature-importance sampling and tree-based method for automatic design flow parameter tuning," in *Proc. ASPDAC*, 2020.

[15] R. Liang, J. Jung, H. Xiang, L. Reddy, A. Lvov, J. Hu, and G.-J. Nam, "FlowTuner: A multi-stage EDA flow tuner exploiting parameter knowledge transfer," in *Proc. ICCAD*, 2021.

[16] H. Geng, T. Chen, Y. Ma, B. Zhu, and B. Yu, "PTPT: Physical design tool parameter tuning via multi-objective bayesian optimization," *IEEE TCAD*, 2022.

[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.

[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.

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

[20] B. Liu, Y. Wang, Z. Yu, L. Liu, M. Li, Z. Wang, J. Lu, and F. V. Fernández, "Analog circuit optimization system based on hybrid evolutionary algorithms," *Integration, the VLSI Journal*, 2009.

[21] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng, and D. Zhou, "An efficient bayesian optimization approach for automated optimization of analog circuits," *IEEE TCAS I*, 2018.

[22] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proc. DAC*, 2020.

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

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

[25] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, 2017.

[26] E. Bisong, *More Supervised Machine Learning Techniques with Scikit-learn*. Apress, 2019.