Automatic Selection of Process Corner Simulations for Faster Design Verification

Michael Shoniker, *Member, IEEE*, Oleg Oleynikov, Bruce F. Cockburn, *Member, IEEE*, Jie Han, *Senior Member, IEEE*, Manish Rana and Witold Pedrycz, *Fellow, IEEE*

***Abstract*—Integrated circuit designs are verified in simulation over a set of process corners, which are combinations of expected transistor properties, power supply voltages, and die temperatures. The simulation time per corner can be long and semiconductor processes can have more than 1000 corners. Simulation is thus a serious bottleneck in design verification. We propose an algorithm that selects the smallest number of process corner simulations that are required to estimate minimum and/or maximum values of the output functions that model circuit behavior. Using our best corner selection algorithm, the required number of process corner simulations is reduced by an average of 79% (a speed-up of 4.71) with respect to a set of 46 output functions from nine industrial benchmark circuits.**

***Index Terms*—Design verification, function approximation, Gaussian processes, process variations, machine learning.**

I. INTRODUCTION

Integrated circuit (IC) designs must be simulated for the range of process properties and operating conditions in a given set of "PVT corners". Each corner is a combination of *process* properties (e.g., the relative switching speed of the transistors), power supply *voltage(s)*, and the die *temperature*. The number of corners has increased to >1000 in some recent technologies [1]-[3]. Each corner simulation can be long, so it is desirable to identify a subset of the corners whose simulation results would still ensure design verification (DV). In DV the output properties (e.g., input-to-output delay, rise & fall times) of a circuit must be verified to stay within specified ranges. This means determining the maximum and/or minimum property values over all corners. This is a combinatorial optimization problem for an expensive-to-evaluate function over a discrete and finite input space. Conventional Response Surface Methodology (RSM) [4]-[5] is inadequate given the large number of PVT properties.

If nothing can be assumed about a function, then finding its optimal values requires a full factorial search of the input space. However, if the form of the function is constrained, it may be possible to find the optimal values after only a fractional-factorial search. For example, Horn showed that if a function is "Lipschitz continuous", it can be possible to safely prune away regions of the input space [6]. We will assume that the functions can be accurately modelled using Gaussian process models (GPMs) [7]. This implies that the functions are sufficiently well correlated and can be accurately represented using standard GPM covariance functions [8]. We will exploit the GPM's ability to provide function value estimates and error estimates. GPMs have been used successfully to solve optimization problems in a wide variety of fields [7], [9].

Similar problems have been considered previously. In [2] McConaghy that mentions the FastPVT tool from Solido Design Automation (Saskatoon, Canada) that uses iterative function approximation based on nonlinear basis models together with simulation to select the best subset of corners. For 108 benchmark circuits, by being able to omit corners, FastPVT produced speed-ups ranging from 43.1× down to 1.0× (no speed-up), with an average speed-up of 11.3× (i.e., omitting 91.15% of the corners). The description of FastPVT in [3] details a method that is different from our approach, where the function estimates and estimated errors from GPMs are combined to select corners. McConaghy also describes tools that that support Monte Carlo (MC) based DV [2][3], where the parameters of individual devices are subject to variation. MC-based DV methods are more general than corner-based methods, but are far more costly computationally and do not scale up easily to large circuits. Li et al. describe an iterative, nested gradient descent method for finding the worst-case corner [10]. It was evaluated for an operational amplifier and shown to produce a speed-up of 21×. However, as noted by McConaghy, many industrial circuits have nonlinear behavior that is not well modeled by linear or quadratic functions. Gradient descent methods are well-known to have difficulties with nonlinear functions with local optima [11]. Sengupta et al. describe how statistical models of device variations together with RSM methods can be used to extract worst-case corners from detailed process data [12]. However, most designers will only have access to a fixed set of corners provided by the foundry or intellectual property (IP) vendor.

The next section presents a problem framework, defines the baseline algorithm, and describes its performance. Sections III, IV and V describe improvements to the initial training set, the termination rule, and the next corner selection rule,

Submitted on November 1, 2016. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Strategic Project Grant STP 447513-13.

M. Shoniker completed an MSc degree in the Dept. of Electrical and Computer Eng. Univ. of Alberta, Edmonton, AB T6G 1H9, Canada.

O. Oleynikov, B. F. Cockburn, J. Han, W. Pedrycz, and M. Rana are with the Dept. of Electrical and Computer Eng., Univ. of Alberta, Edmonton, AB T6G 1H9, Canada (e-mail: {oleyniko|cockburn|jhan8|wpedrycz}@ualberta.ca and manishlistening@gmail.com).

respectively. Section VI identifies inaccuracies in the GPM predictions and then proposes ways to compensate for them. Section VII presents performance evaluation results for the final version of the corner selection algorithm with nine industrial benchmark circuits (46 output functions) provided by Solido. Finally, Section VIII makes some concluding observations and proposes directions for future research.

II. The Baseline Corner Selection Algorithm

A general framework from [13] was adapted in [14][15] for function optimization. Reference [14] describes earlier results that are extended in [15] and summarized here. The total cost tends to be minimized by reducing the number of corner simulations; however, the cost of missing the true optimum of an output function could be ruinous if a violation of correct behavior is missed before production. As in [13], we use an unsupervised machine learning strategy [11]. For convenience and without loss of generality, we seek only the maximum value of each output function over the corner domain: the same algorithm can be readily adapted to find the minimum.

Let X denote the set of all corners, and let $X_i \subseteq X$ denote the set of all corners simulated after $i \geq 1$ iterations of GPM construction. Each X_i is a superset of the previous set X_{i-1}. $G(X_i)$ denotes the GPM that is constructed from X_i and the corresponding simulated function values. We assume that the cost of computing each GPM $G(X_i)$ is much less than the cost of simulating $F(\boldsymbol{x})$ for one corner $\boldsymbol{x}$. The set $\Delta_i = X_{i+1} - X_i$ of corners added to X_i is constructed using both the function estimates $F_{\text{pred}}(\boldsymbol{x})$ and errors $\sigma_{\text{pred}}(\boldsymbol{x})$ produced by $G(X_i)$. The criteria for selecting Δ_i might change as the search progresses to balance the priorities of exploring all regions of the domain versus building confidence that the maximum has indeed been found. An overly greedy heuristic for constructing Δ_i can cause the search to stop at a local maximum [11]. Increasing the size of Δ_i is a way of relaxing the "greed" by forcing the next search increment to be more diverse at the possible cost of performing less informative simulations.

A. Algorithm Description

Objective: Given a set X of corners over $n \geq 1$ PVT parameters and given an expensive simulation model of a function $F(\boldsymbol{x})$, where $\boldsymbol{x} \in X$, find the maximum value $F_{\max}$ of $F(\boldsymbol{x})$ over X and the corner $\boldsymbol{x}_{\max} \in X$ such that $F(\boldsymbol{x}_{\max}) = F_{\max}$.

Step 1: Select the initial training set X_1. This kind of problem is treated in the theory of the design of experiments [5][16]. Following advice from Solido, our initial training set design X_1 of size n^2 includes one modal corner together with $n^2 - 1$ corners selected randomly from the 2^n corners that have extremal values for every parameter. Simulate all corners in X_1 to determine the corresponding values of $F(\boldsymbol{x})$. Set $F(\boldsymbol{x}_{\max?})$ to be the largest of these $F(\boldsymbol{x})$ values, where $\boldsymbol{x}_{\max?}$ is the corresponding corner. Compute $G(X_1)$ from X_1 and the $F(\boldsymbol{x})$ values [7]. We used the package 'scikit-learn" to construct the GPMs [8]. The "absolute exponential" covariance function option was found to give the best results. Set counter i to 1.

Step 2: For each unsimulated corner $\boldsymbol{x} \in X - X_i$, $G(X_i)$ provides an estimate $F_{\text{pred}}(\boldsymbol{x})$ of the function and an estimate $\sigma_{\text{pred}}(\boldsymbol{x})$ of the error in $F_{\text{pred}}(\boldsymbol{x})$ with respect to what the simulated $F(\boldsymbol{x})$ would be [7][8]. Terminate the search and output $F(\boldsymbol{x}_{\max?})$ and $\boldsymbol{x}_{\max?}$ if, for no unsimulated $\boldsymbol{x} \in X - X_i$, does $F_{\text{pred}}(\boldsymbol{x}) + k\,\sigma_{\text{pred}}(\boldsymbol{x})$ exceed the largest simulated $F(\boldsymbol{x}_{\max?})$ found so far in X_i. This is called the k-sigma termination rule. Parameter $k > 0$ is chosen to suit the required confidence that $F_{\max}$ has indeed been found. In the baseline algorithm $k = 3$.

Step 3: Create an increment Δ_i to X_i and let $X_{i+1} = X_i \cup \Delta_i$. Δ_i contains only one unsimulated corner $\boldsymbol{x} \in X - X_i$ that has the greatest value of $F_{\text{pred}}(\boldsymbol{x}) + k\,\sigma_{\text{pred}}(\boldsymbol{x})$. We will call such an $\boldsymbol{x}$ a "worst-case" corner $\boldsymbol{x}_{\text{worst}}$. Simulate all corners in Δ_i and update $F(\boldsymbol{x}_{\max?})$ and $\boldsymbol{x}_{\max?}$ for the largest value of $F(\boldsymbol{x})$ found in X_{i+1}. Compute $G(X_{i+1})$, increment i, and go back to Step 2.

If the $G(X_i)$ produces normally-distributed predictions of $F(\boldsymbol{x})$ with standard deviation $\sigma_{\text{pred}}(\boldsymbol{x})$, then for one unsimulated corner $\boldsymbol{x}_{\text{worst}}$ that just barely passes the k-sigma termination rule with $k = 3$, the simulated value $F(\boldsymbol{x}_{\text{worst}})$ should exceed $F(\boldsymbol{x}_{\max})$ with a probability of close to $Q(k) = Q(3.0) = 0.135\%$, where $Q()$ denotes the Q-function [17]. Thus the k-sigma termination rule should ensure that with probability $\phi(k) = 1 - Q(k)$, the simulated value $F(\boldsymbol{x}_{\text{worst}})$ is indeed less than the greatest value $F(\boldsymbol{x}_{\max?})$ found so far. However, there will likely be $n \geq 2$ unsimulated corners at that time and so a joint probability over those corners must be considered. An exact analysis might be possible (see [18]); however, we found that increasing k by ≥ 0.75 is a simple way of compensating for ≥ 12 equally worst-case corners.

Table 1. Benchmark Circuit Characteristics

Circuit Name	Provided Data Set	Data Set Size	PVT Parameters	Output Functions
shift_reg	Full	1080	5	3
buffer_chain	Full	1800	10	6
bitcell	Full	120	5	2
mux	Fractional	120	8	7
charge_pump1	Fractional	216	8	5
charge_pump2	Fractional	324	8	5
sense_amp	Fractional	120	10	7
bias_gen	Fractional	120	3	10
op_amp	Fractional	120	6	1

Table 2. PVT Parameters for the "shift_reg" Benchmark Circuit

Process	ss, sf, fs, ff, tt	**vvcc (V)**	3.2, 3.3, 3.4
Temp. (C)	-50, -25, 0, 27, 50, 75, 100, 125	**vvdd (V)**	1.4, 1.5, 1.6
		vvref (V)	1.6, 1.65, 1.7

B. Simulation Results for the Baseline Algorithm

We evaluated the baseline algorithm using nine benchmark circuits, which provided 46 output functions, see Table 1. The simulated outputs at each corner, but not the circuit netlists, were provided by Solido from a mix of typical and challenging industrial circuits. Circuit "shift_reg", presumably a shift register, was especially interesting. Its three output functions ("delay", "fall time" & "rise time") proved difficult to learn. This circuit had five PVT parameters, see Table 2. Following standard practice [19], the "process" parameters give the relative speed of the N- and P-type transistors: T(ypical)T, S(low)S, F(ast)S, SF, FF). These five values conflate some independent variations affecting the two types.

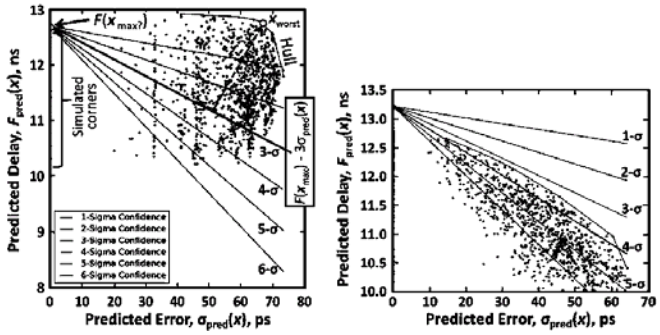

Fig. 1. Convex hull plots for the "delay" Output of "shift_reg" after (left) 35 and (right) 100 PVT Corners have been Simulated.

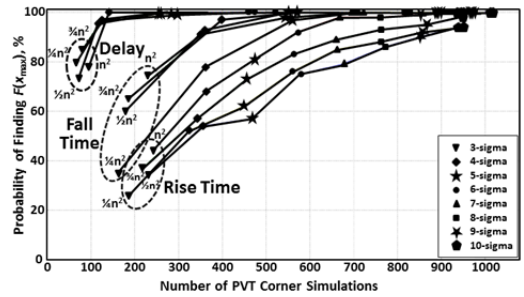

Fig. 2 Probability of finding F_{max} with increasing numbers of corner simulations for outputs "delay", "fall time" and "rise time" in benchmark circuit "shift_reg", assuming initial training set sizes of 49 (n^2), 36 (≈ ¾ n^2), 24 (≈ ½ n^2), and 12 (≈ ¼ n^2), with 100 random sets for each plotted point.

We thus remapped the five conventional process values to two separate three-valued process parameters. This changed the "full factorial" datasets in Table 1 to "fractional factorial". This change had little effect, slightly slowing down convergence on average but increasing the accuracy of termination.

Consider the 2-D scatter plots of the function values versus the predicted errors in Fig 1. Fig. 1(left) shows the plot after 35 simulations of the "delay" output. The 35 simulated outputs appear as dots on the vertical axis since their uncertainty is near-zero. The maximum simulated value, $F(\boldsymbol{x}_{max?})$, found so far is the uppermost dot on the vertical axis. The predicted values $F_{pred}(\boldsymbol{x})$ appear as dots to the right of the vertical axis with errors $\sigma_{pred}(\boldsymbol{x})$. The convex hull around the upper right side of the predicted points, employed later, was efficiently computed using a modified version of Graham's scan algorithm [20]. György and Kocsis also used hull plots [21].

The so-called "*k*-sigma" termination rule is that all predicted values $F_{pred}(\boldsymbol{x})$ must be less than $F(\boldsymbol{x}_{max?}) - k\ \sigma_{pred}(\boldsymbol{x})$ for some suitable $k > 0$. This rule appears in Fig. 1 as six lines, corresponding to $k = 1, \ldots, 6$, that fall down and to the right with a vertical intercept of $F(\boldsymbol{x}_{max?})$ and with slopes of -1, …, -6, respectively. The hull plots clarify how the algorithm selects corners to simulate. In Fig. 1(left), the next corner that will be selected, $\boldsymbol{x}_{worst}$, has the greatest value of the cost function $F_{pred}(\boldsymbol{x}) + k\ \sigma_{pred}(\boldsymbol{x})$. $\boldsymbol{x}_{worst}$ has the greatest perpendicular distance from the *k*-sigma rule. As the algorithm progresses, predicted values are replaced with simulated values and the corresponding dots move left to the vertical axis. Fig. 1(right) shows the scatter plot after 100 simulations for "delay". The dots for the remaining predicted values shift downwards and to the left. In Fig. 1(right) the convex hull has fallen below the 1-, 2- and 3-sigma lines. Our confidence that we have found the F_{max} (i.e, $F(\boldsymbol{x}_{max?}) = F_{max}$) increases as the hull moves down past rules with increasing values of *k*.

The corner selection rule must minimize the probability that termination occurs before F_{max} has been found. Deriving this probability is impractical but one can approximate it by re-running the algorithm for, say, 100 randomly-generated choices of X_1 to determine the fraction of those runs that found F_{max}. Fig. 2 shows the results of experiments for the "delay", "fall time" and "rise time" outputs of "shift_reg". Each symbol in Fig. 2 corresponds to 100 runs with equal-sized initial training sets with different random choices of the extremal corners. The training sets had sizes 49 (n^2 corners), 36 (≈ ¾ n^2), 24 (≈ ½ n^2), and 12 (≈ ¼ n^2). The points for the same functions and X_1 sizes have been linked up to show the increasing "probability" of finding F_{max} as more corners are simulated. For each curve the leftmost point shows the number of simulations when the 3-sigma termination rule was first satisfied; subsequent points show when the 4, 5, …, 10-sigma termination rules were first satisfied. For "delay" when the 4-sigma termination rule was first reached going from 49 to 142 corner simulations, all 100 runs found F_{max}. For this output, the 3-sigma termination rule is clearly insufficient and at least the 6-sigma rule should be used. Had the 6-sigma termination rule been used to verify "delay", the number of simulations would have fallen from 1080 to 461, for a reduced effective "speed-up" of 2.34×. Fig. 3 also illustrates the greater difficulty of "fall time" and "rise time" compared with "delay". Function "fall time" required 571 simulations to satisfy the 5-sigma termination rule, when all 100 runs found F_{max}. The speed-up would then be 1080/571 = 1.89×. For "rise time", after 935 corners only 98 of the runs found F_{max}.

III. Improvements to the Initial Training Set

The baseline algorithm uses an initial training set X_1 of size n^2, but it is unclear if this is a good choice for the 46 output functions or for any other circuit. For some circuits the n^2 size is clearly too big. For example, the "sense amp" benchmark circuit has 10 PVT parameters and 120 corners. An initial training set of size $10^2 = 100$ would not allow significant speed-up. We thus investigated the effects of reducing the size of X_1. A smaller initial training set might allow the termination rule to be reached with fewer corner simulations since the learning algorithm can start adapting the GPM sooner.

The results in Fig. 2 show how performance varied for initial training sets of size n^2, ¾ n^2, ½ n^2, and ¼ n^2. For the "delay" output, all four sizes of X_1 converged onto F_{max}, with the n^2 size converging a bit faster than the three smaller sizes. For the "fall time" output, all four set sizes converged in roughly the same number of corner simulations, although the smallest initial training set size (¼ n^2) lagged the three other sizes for most of their runs. For the hardest "rise time" output, the algorithm failed to find the true F_{max} with 100% probability (over the 100 trials) after 895 corner simulations when the initial training set size was smaller than n^2.

For the functions "delay", "rise time" and "fall time" it appears that shrinking X_1, while retaining the same increment size of one new corner simulation per iteration, did not lead to faster or more reliable convergence onto the function's F_{max}. The benefit of using a larger X_1 might be that it forces the algorithm to invest a larger number of simulations in a less greedy initial exploration of the full input domain.

We also investigated the convergence performance using a much smaller X_1 whose size is determined by the rule max(0.01m, 2n), where m is the number of corners. This rule avoids the situation where a set X_1 of size n^2 would be an overly large fraction of all corners. A second improvement was made to X_1 by selecting the extremal corners iteratively where each new corner is at a maximum Manhattan distance from all extremal corners selected earlier. The values within each PVT parameter domain were mapped to integer sequences of the form 0, 1, …, max_value to provide the necessary grid points in the n-dimensional PVT corner domain.

IV. Improvements to the Termination Rule

We investigated the causes for the algorithm's occasional failures to find F_{max}. Premature termination was found to be often caused by failure to detect F_{max} at an unsimulated corner when that corner was adjacent, in a Manhattan sense, to the final $F(\boldsymbol{x}_{max?})$. The predicted errors produced by the GPM were evidently overly small for unsimulated corners near $\boldsymbol{x}_{max?}$.

We improved the reliability of the termination rule by including a multiplier factor $E(\boldsymbol{x})$ that magnifies the predicted error $\sigma_{pred}(\boldsymbol{x})$ for corners near $\boldsymbol{x}_{max?}$ to make it more likely that those corners will be simulated. After some experimentation, we defined $E(\boldsymbol{x})$ to be 1.25 when an unsimulated corner $\boldsymbol{x}$ differs from $\boldsymbol{x}_{max?}$ by one Manhattan step, and to have the value 1.15 when $\boldsymbol{x}$ differs from $\boldsymbol{x}_{max?}$ by two Manhattan steps; otherwise, $E(\boldsymbol{x})$ has value 1.0 (no enhancement). This three-level $E(\boldsymbol{x})$ produced more reliable termination compared to two-level $E(\boldsymbol{x})$'s that we considered. Four-level $E(\boldsymbol{x})$'s greatly increased the run time without improving termination.

V. Improvements to the Next Corner Selection Rule

After computing the first GPM from the initial training set X_1, the baseline algorithm enlarges X_i at each iteration step $i \geq 1$ by adding one unsimulated corner $\boldsymbol{x}_{worst}$ that has the greatest value of $F_{pred}(\boldsymbol{x}) + k\ \sigma_{pred}(\boldsymbol{x})$, where $k > 0$ is the sigma confidence parameter. Adding only one corner allows the GPM to be updated sooner so that F_{max} might be found faster. However, faster convergence was actually obtained when multiple roughly equally worst-case corners are added [15].

Table 3 shows the successive improvements that were made to the algorithm to this point. The columns headed "3-σ", …, "6-σ" show the number of corner simulations required to satisfy the 3-, …, 6-sigma termination rules, respectively, when the predicted errors $\sigma_{pred}(\boldsymbol{x})$ are used without enhancement and where the X_i's grow by one $\boldsymbol{x}_{worst}$ at each step. Asterisks indicate cases where the F_{max} was not found in all 100 trials (but still found in most trials). The next column gives the number of corners to reach termination after the 3-level $E(\boldsymbol{x})$ is applied to $\sigma_{pred}(\boldsymbol{x})$. The last column shows the number when X_{i+1} is constructed by adding all of the corners in the ith convex hull to X_i. Note that F_{max} was correctly found for all three output functions for the last version of the algorithm. The cost (in number of corner simulations) fell for "delay" and "fall_time" but increased for "rise_time".

Table 3. Number of Corner Simulations Required to Reach Termination
Note: Asterisks indicate where F_{max} was reached in most but not all 100 trials.

Output Function	3-σ k=3	4-σ k=4	5-σ k=5	6-σ k=6	6-σ 3-lev. $E(x)$ 1-corner	6-σ 3-lev. $E(x)$ Full hull
delay	95*	141	283	461	511	290 (3.72×)
fall_time	229*	399*	571	710	710	300 (3.60×)
rise_time	242*	362*	475*	572*	539*	816 (1.32×)

Table 4. Single-Output Function Performance for the Benchmark Circuits

Circuit Name	Initial Training Set Size	Corners to Find F_{max}	Corners Until Term.	Min., Max. and Mean Speed-Ups
shift_reg	14 of 1080	84-306	478-1047	1.03, 2.25, 1.46
buffer_chain	20 of 1800	29-45	99-365	4.93, 18.18, 13.32
bitcell	10 of 120	12	26	4.61, 4.61, 4.61
mux	16 of 120	16-69	18-113	1.06, 6.67, 2.90
charge_pump1	16 of 216	16-23	38-50	4.32, 5.68, 5.02
charge_pump2	16 of 324	16-23	53-71	4.56, 6.11, 5.31
sense_amp	20 of 120	20	35-114	1.05, 3.53, 2.37
bias_gen	10 of 120	10	36-41	2.93, 3.33, 3.18
op_amp	12 of 120	12	46	2.61, 2.61, 2.61

VI. Improvements to the Function Model

For most of the circuits, the algorithm had significant speed-up. However, for some functions convergence was slow and/or termination was premature. The problem arises from inaccuracy in both the $F_{pred}(\boldsymbol{x})$ and $\sigma_{pred}(\boldsymbol{x})$ values. If the distributions of $F_{pred}(\boldsymbol{x})$ are truly Gaussian with standard deviations given by $\sigma_{pred}(\boldsymbol{x})$, then 99.7% of the time the absolute error should be less than or equal to $3\sigma_{pred}(\boldsymbol{x})$. We found that for randomly chosen training sets of increasing size, the $\sigma_{pred}(\boldsymbol{x})$ values were increasing worse underestimates.

Using cross-validation [11] we computed a boosting factor, β, that corrects the predicted errors $\sigma_{pred}(\boldsymbol{x})$ so that predicted values $F_{pred}(\boldsymbol{x})$ will just lie within ± 3 β $\sigma_{pred}(\boldsymbol{x})$ of the actual $F(\boldsymbol{x})$ values. The product $\beta \times \sigma_{pred}(\boldsymbol{x})$ can then replace $\sigma_{pred}(\boldsymbol{x})$ in the termination and next corner selection rules. β can be recomputed using cross-validation for each X_i. As each simulated corner appears as a test data point for a learned model in cross-validation, the ratio $\beta_j = |\ F(\boldsymbol{x}) - F_{pred}(\boldsymbol{x})\ | / 3\ \sigma_{pred}(\boldsymbol{x})$ is calculated. Defining an overall boosting factor β to be the maximum value of all the β_j ratios improved the behavior of the selection algorithm. However, it is expensive to re-compute a new β for every X_i. With little loss in effectiveness, β can be computed first using cross-validation with X_1, then recomputed after every subsequent multiple of 10% of the input space, and projected linearly elsewhere.

VII. Evaluation of the Improved Algorithm

A final version of the algorithm was evaluated against 46 functions from the 9 circuits in Table 1. The initial set X_1 had

size $q = \max(0.01m, 2n, 10)$ and included one "typical" corner plus q-1 spaced-out extremal corners. β is calculated using 10-fold cross-validation with extrapolation. The same three-level factor $E(\boldsymbol{x})$ is used to boost $\sigma_{pred}(\boldsymbol{x})$ for corners $\boldsymbol{x}$ that are near the present $\boldsymbol{x}_{max?}$. The training set increment includes all points on the convex hull and the termination rule used $k = 4$.

Table 4 summarizes results that are reported in greater detail in [15]. The average speed up over all 46 functions was 4.71, which is a saving of 79%, that is, (1 – 1/4.71) × 100%, below a full factorial set. However, the speed-ups in the last column had significant variability across circuits and across the functions in each circuit. Circuit "buffer_chain" had six relatively easy-to-learn outputs, which allowed for speed-ups ranging from 4.93× to 18.18× with a mean of 13.32× (i.e., a 92% reduction in corners simulated).

The hardest benchmarck was "shift_reg", with a mean speed-up of 1.46× (a 31.5% reduction in corner simulations). Its "delay" function required 478 corners to reach termination (a 2.25× speed-up). Although the F_{max} was encountered at the 84th corner, 394 additional simulations were required to reach termination. The "fall_time" function was the hardest. Its F_{max} was found after 163 corner simulations, but termination required 47 simulations (a 1.03× speed-up). Only 33 simulations were avoided, a 3.06% reduction below 1080. The "rise_time" function was slightly less difficult. The F_{max} was found after 306 simulations with termination occurring after 988 simulations (a 1.09× speed-up). Only 92 corner simulations were avoided, an 8.5% reduction below full factorial.

We identified two scenarios where the algorithm converged incorrectly. A discontinuity, like an isolated spike or a narrow ridge, was hard to model as a GPM. The "fall_time" and "rise_time" functions indeed have ridges, with three adjacent corners having near-equally high values. The "sen_dip_pctg" function of "sense_ amp" has a broad plateau region where the function varies irregularly in value by only 0.14%.

VIII. Conclusions and Future Research

We considered the problem of automatically minimizing the number of corners that must be simulated to determine the maximum (or minimum) value of an output function that describes behavior of an arbitrary circuit. We proposed an algorithm that iteratively builds up a Gaussian Process Model for the function being verified. The GPM produces estimates for the unsimulated function values and errors for those values; these estimates are combined to select the corners to simulate next as well as to determine when to stop the search.

Nine benchmark circuits, with 46 functions, were considered. The algorithm always successfully found the maximum function value, but the reduction in the number of corner simulations varied considerably. The algorithm produced an average speed-up of 4.71×, which is a reduction of 79% in the number of simulations. However, for some circuit output functions little or no speed-up could be obtained over a full-factorial search. FastPVT [2] produced a speed-up of 11.3× for a different set of benchmarks. It is unclear how robust this speed-up is to the choice of initial training set.

Three priorities in future work are to increase the number of benchmark circuits, to better understand what causes some output functions to be so hard to learn using the iterative GPM-based approach, and to coordinate the search for multiple functions from the same circuit. Additional function models (e.g., [22]) will be considered in addition to GPMs.

Acknowledgment

The authors thank Solido Design Automation Inc. for their technical advice and assistance during the project.

References

[1] K. L. Kuhn et al., "Process technology variation," *IEEE Trans. Electron Devices*, vol. 58, no. 8, pp. 2197-2208, Aug. 2011.

[2] D. De Jonghe et al., "Advances in Variation-Aware Modeling, Verification, and Testing of Analog ICs," *Design, Automation & Test in Europe Conf.*, Dresden, Mar. 12-16, 2012, pp. 1615-1620.

[3] T. McConaghy, K. Breen, J. Dyck and A. Gupta, *Variation-Aware Design of Integrated Circuits: A Hands-on Field Guide*, New York: Springer Science+Business Media, 2013.

[4] G. E. P. Box and K. B. Wilson, "On the experimental attainment of optimal conditions," *J. of the Royal Statistical Society*, Series B, vol. 13, no. 1, pp. 1-45, 1951.

[5] R. H. Myers, D. C. Montgomery, C. M. Anderson-Cook, *Response Surface Methodology: Process and Product Optimization using Experiments*, 3rd ed., Hoboken, NJ: John Wiley & Sons, 2009.

[6] M. Horn, "Optimal algorithms for global optimization in case of unknown Lipschitz constant," *J. of Complexity*, vol. 22, no. 1, pp. 50-70, Feb. 2006.

[7] C. E. Rasmusssen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, Cambridge, MA: MIT Press, 2006.

[8] F. Pedregosa et al., Scikit-learn: Machine Learning in Python, *J. of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[9] G. Su, "Accelerating Particle Swarm Optimization Algorithms Using Gaussian Process Machine Learning," *Proc. Int. Conf. on Computational Intelligence and Natural Computing (CINC)*, Wuhan, China, June 6-7, 2009, pp. 174-177.

[10] M. Li, D. Zhou, and X. Zeng, "NIO: A Fast and Accurate Verification Method for PVT Variations," *Proc. 12th IEEE Int. Conf. Solid-State and Integr. Circ. Technol. (ICSICT)*, Guilin, China, Oct. 28-31, 2014, 3 pp.

[11] E. Alpaydin, *Introduction to Machine Learning*, 3rd ed., Cambridge, MA: MIT Press, 2014.

[12] M. Sengupta et al., "Application-specific worst case corners using response surfaces and statistical models," *IEEE Trans. CAD*, vol. 24, no. 9, pp. 1372-1380, Sept. 2005.

[13] D. R. Jones, M. Schonlau, W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *J. of Global Optimization*, vol. 13, pp. 455-592, 1998.

[14] M. Shoniker, B.F. Cockburn, J. Han and W. Pedrycz "Minimizing the number of process corner simulations during design verification," in *Design, Automation & Test in Europe*, Dresden, 2015, pp. 289-292.

[15] M. Shoniker, "Accelerated Verification of Integrated Circuits Against the Effects of Process, Voltage and Temperature Variations," M.Sc. thesis, Dept. Elect. Comp. Eng., U. Alberta, Edmonton, Canada, 2015.

[16] T. P. Ryan, *Modern Experimental Design*, Hoboken, NJ: Wiley, 2007.

[17] N. A. Weiss, *Introductory Statistics*, 10th ed., New York: Pearson, 2015.

[18] C. Visweswariah et al., "First-Order Incremental Block-Based Statistical Timing Analysis," *Proc. 41st Design Automation Conference (DAC)*, San Diego, CA, June 7-11, 2004, pp. 331-336.

[19] N. H. E. Weste and D. Harris, *CMOS VLSI: A Circuits and Systems Perspective*, 4th ed, Boston, MA: Pearson Addison-Wesley, 2010.

[20] R. L. Graham, "An Efficient Algorithm for Determining the Convex Hull of a Planar Point Set," *Inform. Process. Lett.*, v. 1, pp. 132-3, 1972.

[21] A. Gyorgy and L. Kocsis, "Efficient multi-start strategies for local search algorithms," *J. Artif. Int. Res.*, vol. 41, pp. 407-444, May 2011.

[22] Y. Tenne and S. W. Armfield, "A Novel Evolutionary Algorithm for Efficient Minimization of Expensive Black-box Functions with Assisted-Modelling," *IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, Canada, July 16-21, 2006, pp. 3219-3226.