Circuit Performance Classification With Active Learning Guided Sampling for Support Vector Machines

Honghuang Lin and Peng Li, *Senior Member, IEEE*

***Abstract*—Leveraging machine learning has been proven as a promising avenue for addressing many practical circuit design and verification challenges. We demonstrate a novel active learning guided machine learning approach for characterizing circuit performance. When employed under the context of support vector machines (SVMs), the proposed probabilistically weighted active learning approach is able to dramatically reduce the size of the training data, leading to significant reduction of the overall training cost. The proposed active learning approach is extended to the training of asymmetric SVM classifiers, which is further sped up by a global acceleration scheme. We demonstrate the excellent performance of the proposed techniques using four case studies: 1) dc/dc converter ripple noise analysis; 2) phase-locked loop lock-time verification; 3) reliability analysis of a ring oscillator with respect to process variations and initial conditions; and 4) prediction of chip peak temperature using a limited number of on-chip temperature sensors.**

***Index Terms*—Active learning, circuit performance classification, support vector machine (SVM).**

I. Introduction

UNDERSTANDING circuit performances' dependencies on key design, process, and operating condition parameters are a central issue in many phases of IC development. For instance, design equations or circuit performance models that capture specifications' dependency on design parameters are essential for guiding design optimization.

However, as a by-product of design complexity increase and technology scaling, design, process parameters, and operating conditions interact with design performance and operation in an increasingly complex manner. Optimizing design performance and safeguarding design robustness becomes a significant challenge.

In this light, the ability in characterizing circuit performance in the complex design/technology/operating parameter space is key to design, verification, and test. In practice, performance characterization process often entails collecting and processing large volumes of simulation or measurement data, which can be extremely costly and time consuming. To address the above challenge, we leverage a specific type of machine learning techniques, support vector machines (SVMs) [1], [2], for efficient characterization of design performance.

Machine learning has been adopted in electronic design automation (EDA) research in the past. Voting-based methods are used in [3] and [4] to train analog performance models parameterized in design parameters. In [4] and [5], one-class SVM is adopted to represent analog circuit performance and perform outlier analysis for cost reduction of delay test. In [6] and [7], regression techniques are used to analyze circuit reliability and rank design features that contribute to unmodeled systematic timing effects. Focusing on a somewhat different problem, Singhee and Rutenbar [8] combined the extreme value theory and machine learning (e.g., SVM) for yield estimation of static random access memories (SRAMs). The same problem is approached by several other authors through fast Monte Carlo importance sampling techniques or boundary searching based on non-Monte Carlo methods [9]–[11].

SVM is well known for its capability of handling nonlinear problems. Compared to other classification techniques in the machine learning domain, SVM tends to generate a sparser solution from an evenly sampled training data set. As a useful toolbox in the EDA community, leveraging such sparsity to reduce the intense need in sampling has been largely left untouched, which is the key focus of this paper. We propose an optimized probabilistic active learning guided SVM approach to train high-quality circuit performance classifiers with significantly reduced cost. As shown in Fig. 1, active learning is a class of powerful techniques that can be leveraged to build "intelligence" into the sampling step of classifier training [12]–[14]: only promising instances that are expected to improve the performance of the classifier are selected for query (e.g., circuit simulation).

To select the optimal query instance at a time, we rank the candidate instances according to a probabilistically weighted goodness metric that is computed by rigorously evaluating potential reduction of version space if the instance were queried. As such, our active learning scheme intelligently selects query instances which are expected to act as support vectors throughout the iterative classifier training process, hence avoids to committing wasteful queries and significantly

Manuscript received May 25, 2014; revised October 8, 2014 and December 29, 2014; accepted February 16, 2015. Date of publication March 16, 2015; date of current version August 18, 2015. This work was supported in part by the National Science Foundation under Grant 1117660, and in part by the Semiconductor Research Corporation (SRC) task # 1836.128 through The University of Texas at Dallas' Texas Analog Center of Excellence (TxACE). This paper was recommended by Associate Editor Y. Shin.

The authors are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: linhh@tamu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2015.2413840

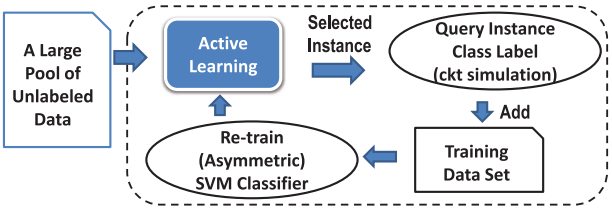
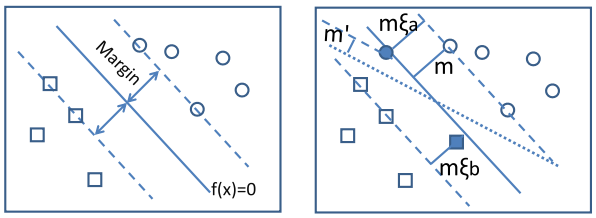


Fig. 1. Active-learning guided SVM.



(a) (b)

Fig. 2. (a) Hard margin SVM. Circles and squares are instances of the two classes. $f(x) = 0$ is the separating hyperplane. (b) Differences between soft and hard margins SVM. The dotted line is the hyperplane of hard margin SVM with margin m'. The solid line is the hyperplane of soft margin SVM with larger margin m. The solid circle and square violate the margin m with slack variable ξ_a and ξ_b, respectively.

reduces the required training data and the overall training cost.

For many binary classification problems in circuit design, certain degree of asymmetry exists between the two classes. For instance, in design verification, it is practically relevant to answer the following question: how to ensure designs passing the verification are very likely to be actually acceptable, while allowing some of the acceptable designs to be rejected. To ensure the conservativeness in verification, the error on predictions of acceptance shall be more heavily minimized to avoid overly optimistic prediction of design robustness. To this end, we extend the proposed probabilistic active learning to the case of asymmetric SVMs by generalizing the concept of version space reduction and evaluating the biased safety level. Finally, we present a global acceleration scheme to further improve the convergence of asymmetric SVMs.

The presented ideas are rather general. We demonstrate their application by conducting four classification case studies: 1) ripple noise analysis of LC-based dc/dc converters; 2) lock-time verification of charge-pump phase-locked loops (PLLs); 3) reliability analysis of ring oscillators; and 4) prediction of peak chip temperature based on a limited number of on-chip temperature sensors. For these applications, our techniques have led to one order of magnitude of efficiency improvement.

II. SVM

SVM [1], [2] is a useful supervised learning algorithm in solving binary classification problems. Given a set of training samples, the technique constructs a discriminant function as a classifying hyperplane in the input space. Its training process can be solved as a quadratic programming optimization problem. The objective of the optimization is to find the structural optimal hyperplane that separates the training data with largest margin. Such objective is called structural risk minimization and provides promising performance in many situations.

In practice, training samples are often not linearly separable, especially in circuit related applications. To achieve a nonlinear classifier, SVM projects the input data into a higher dimensional space, which is called feature space, and then tries to linearly separate the samples in the feature space. The higher dimensional hyperplane may have a nonlinear projection in the input space, serving as a nonlinear classifier for the input data.

Denote the ith sample in the training data set as

$$(x_i, y_i), y_i \in \{-1, +1\}, i = 1, 2, 3, \ldots, n \tag{1}$$

which consists of the input vector x_i and the corresponding class label y_i. Let $\phi(x)$ be the mapping function that maps any input vector x from $\mathbb{R}^n$ to $\mathbb{R}^m$. SVM defines the discriminant function of the classifier as

$$f(x) = w \cdot \phi(x) + b \tag{2}$$

where w is a vector with m entries. An unlabeled x will be classified as a positive or negative instance if $f(x) > 0$ or $f(x) < 0$, respectively. The separating hyperplane in the feature space is defined by $f(x) = 0$. Instances closest to the hyperplane are defined as support vectors. The distance from any support vector to the hyperplane is called margin, which should be maximized during the training process.

The primal form of SVM is defined as follows:

$$\min_{w,b} \frac{\|w\|^2}{2} \tag{3}$$

$$\text{subject to } y_i(w \cdot \phi(x_i) + b) \geq 1. \tag{4}$$

Equality of (4) holds when x_i is a support vector. Based on the definition and constraints described in (4), the margin can be computed by

$$m = \frac{1}{\|w\|} \tag{5}$$

and thus minimizing the objective function is actually maximizing the margin [see Fig. 2(a)].

To solve the equivalent optimization problem of the SVM model, Lagrange multipliers are often employed, and the dual form of the SVM problem is derived as

$$\min_{\alpha_i} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \cdot \phi(x_i) \cdot \phi(x_j) \tag{6}$$

$$\text{subject to } \alpha_i \geq 0 \tag{7}$$

and

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{8}$$

where Lagrange multipliers α_i are the new model parameters. The support vectors can then be defined as training vectors

whose corresponding α_i is nonzero. And the original w in (2) can be expressed as

$$w = \sum_{i=1}^{n} \alpha_i y_i \cdot \phi(x_i). \tag{9}$$

From the dual problem, we can find that only inner products of vectors in the feature space are involved in computation. Thus, we can simply define a kernel function $K(x_1, x_2) = \phi(x_1) \cdot \phi(x_2)$ to implicitly describe the mapping toward the feature space while solving the optimization problem instead of finding an exact mapping function $\phi(x)$.

One of the widely used kernel functions is called radial basis functions (also known as Gaussian kernel) [2]

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}. \tag{10}$$

The usage of kernel functions, the form of the discriminant function, and the structural risk minimization make SVM distinguished from other classification techniques, since the optimal solution may be sparse in the context that it can be represented by just a small portion of the samples, i.e., support vectors.

Leveraging the kernel (10), the biasing term b in the discriminant function is often set to be zero for simplicity. With $b = 0$, the constraint (8) will be freed, leaving (7) be the only constraints of the dual problem. In this paper, we use (10) as the default kernel and adopt the unbiased hyperplane by setting $b = 0$ in order to facilitate active learning [13] (details in later sections).

Overfitting is one of the common obstacles in supervised learning techniques. The fact is that the "strictly learned" discriminant function from a large amount of training data is often outperformed by well regularized discriminant functions with some extent of relaxation. To balance the overfitting and underfitting, a modified SVM called soft margin SVM is proposed in [1] and [2].

Rather than satisfying all the constraints defined by the training data set, soft margin SVM uses slack variables and a cost factor to tradeoff the training error and margin. The primal form of soft margin SVM can be posed as follows:

$$\min_{w,\xi} \ \frac{\|w\|^2}{2} + C \sum_{i=1}^{n} \xi_i \tag{11}$$

$$\text{subject to} \ \ y_i(w \cdot \phi(x_i)) \geq 1 - \xi_i, \xi_i \geq 0 \tag{12}$$

where C is the constant cost coefficient defined before the training and ξ_i is the slack variable representing the margin violation of the ith training example. And the margin [as shown in Fig. 2(b)] is defined as

$$m = \min_i \frac{y_i f(x_i) + \xi_i}{\|w\|} = \frac{1}{\|w\|}. \tag{13}$$

Similarly, by applying Lagrange multipliers to the primal problem, one can derive the dual problem with exactly the same objective function as (6) but with different constraints

$$0 \leq \alpha_i \leq C. \tag{14}$$

In this case, support vectors can still be defined as vectors with $\alpha_i > 0$. For those vectors with $\alpha_i < C$, they locate right on the boundaries of the margin. For those vectors with $\alpha_i = C$, they are outliers that violates the margin, sometimes may even be misclassified samples.

III. Active Learning Guided SVM

The goal of circuit performance classification is to predict if a circuit with uncertain parameters (i.e., design parameters, process variations, or working conditions) will meet the performance specification or not. Given sufficient training data, an SVM classifier can be trained for accurate prediction. However, expensive circuit simulation usually makes it infeasible to obtain a large volume of samples to train an accurate SVM.

To reduce the intense need of training data, a group of techniques called semi-supervised active learning [15] is developed by leveraging the spatial information (e.g., clustering based on [16]) or the sampling history (e.g., significance space construction in [17]). It only works for problems with finite candidates and thus is not applicable to analog circuits whose sampling space is usually infinite. Supervised active learning uses a clearer strategy to select samples: hypothesis space reduction (e.g., agnostic active learning [18]) or SVM oriented version space reduction [13], [19]. In this domain, the concern lies on the efficiency of the algorithm. However, in circuit application, since simulations can be extremely expensive, we need to optimize the quality of the selected sample as well. Therefore, in this section, we propose active learning for high quality and fast convergence. Another active learning scheme for conservative prediction is proposed in the next section.

A. Version Space

Pool-based active learning [12] is a method that starts with an initial training data set and a pool of unlabeled data. The learner can query the class labels of instances in the pool and add them to the training data set.

Different active learning strategies have been developed in different scenarios for the selection of candidates from the pool. In the EDA domain, for example, [4] employs active learning to generate a balanced, in terms of the number of different types of samples, training data set. Another method proposed in [6] develops active learning strategy to reduce the need of expensive aging simulations in building regression models for circuit reliability analysis.

A querying strategy based on version space introduced by Tong and Koller [13] is an effective method for active learning with SVM. With this strategy, active learner would start with a set of all the possible separating hyperplanes, i.e., the version space, and then choose the next query to shrink this set as much as possible until the size of the set is small enough. Then, the optimal hyperplane of this small set could be used as an accurate approximation of the real-separating hyperplane (more details in the next section).

Version space V is defined as the set of all the hyperplanes in the form of $f(x) = w \cdot \phi(x) = 0$ that completely

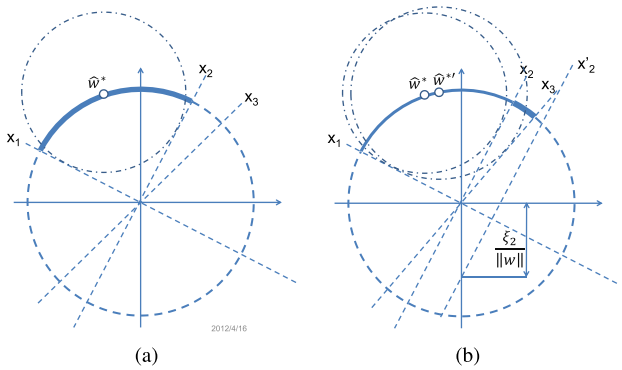


Fig. 3. Version space in 2-D W space. (a) x_1–x_3 are cutting hyperplane associated with training data. x_1 and x_2 are support vectors. The solid arc is the version space. (b) $\hat{w}^*$ and version space between x_1 and x_2 corresponds to hard margin SVM. In soft margin SVM, hyperplane of x_2 is shift to x_2' with a vertical offset of $\xi_2/\|w\|$. The version space is enlarged by the arc between x_2 and x_3. $\hat{w}^{*\prime}$ corresponds to the optimal classifier of soft margin SVM. x_3 becomes the support vector as well.

separate the training data in the feature space. Since the normalization of w

$$\hat{w} = \frac{w}{\|w\|} \tag{15}$$

will not change the hyperplane, we can define the version space as

$$V = \left\{ \hat{w} \in W \middle| \|\hat{w}\| = 1, y_i\big(\hat{w} \cdot \phi(x_i)\big) > 0 \ \forall i \right\} \tag{16}$$

where the parameter space W has the same dimension of the feature space F. According to the duality between the parameter space W and feature space F [2], every training sample x_i has a corresponding hyperplane $\phi(x_i) \cdot w = 0$ in W that cuts off a part of the hypersphere $\|\hat{w}\| = 1$. An example in a 2-D parameter space is shown in Fig. 3(a). Each training data is actually a constraint in the optimization problem, partially defining the set of all the feasible solutions. As a result, all the infeasible $\hat{w}$ will be cutoff by any training sample they violate and the surface on the hypersphere that remains at last will be the version space.

Due to the structural risk minimization in the SVM model, the corresponding $\hat{w}^* \in V$ of the optimal classifier produced by the hard margin SVM will be the center of a sphere which is tangent to the cutting hyperplanes in W associated with support vectors [see Fig. 3(a)]

$$\hat{w}^* = \arg\max_{\hat{w} \in W} \min_i \big\{ y_i\big(\hat{w} \cdot \phi(x_i)\big) \big\}. \tag{17}$$

Here, $\|\hat{w}\|$ is normalized instead of the normalization of $|f(x_i)|$ for support vectors in the hard margin SVM problem (3), (4). Therefore, $\hat{w}^*$ is the normalization of the optimal separating hyperplane represented by w^*, and the distance between support vectors and this hyperplane is the margin m in (5).

Note that (16) is defined under the condition that all the training data are linearly separable in the feature space. However, since most circuit variables (i.e., voltage, current, etc.) are continuous and the characteristics are often nonlinear, samples in circuit application are usually not completely separable. Thus, we employ soft margin SVMs instead of hard margin SVMs in our applications and propose another definition of version space V for the soft margin SVM

$$V = \left\{ \hat{w} \in W \middle| \|\hat{w}\| = 1, y_i\big(\hat{w} \cdot \phi(x_i)\big) > -\frac{\xi_i}{\|w\|} \ \forall i \right\} \tag{18}$$

where w and ξ_i are constants produced by the training process of soft margin SVM (11), (12) with given cost coefficient C. As mentioned in the previous section, the cost coefficient C provide a tradeoff mechanism between the structural risk and constraint slackness. In practice, such mechanism often provides much better regularization, hence it is reasonable to assume that the slackness ξ_i can also make the version space more resistant to the overfitting problem.

As shown in Fig. 3(b), a nonzero slack variable ξ_i moves the corresponding cutting hyperplane vertically, relaxing the constraint defined by x_i. Therefore, compared to the version space of hard margin with the same training data, soft margin SVM enlarges the size of the version space as well as the margin of the classifier.

Alternatively, from the perspective of the dual problem and the kernel method, if we define $\hat{\alpha}_i = \alpha_i y_i$, by substituting (9) into (2), the form of all the possible discriminant function can be defined as

$$f_d(x) = \sum_{i=1}^{n} \hat{\alpha}_i \cdot K(x_i, x). \tag{19}$$

Then, the equivalent hard margin version space can be redefined as

$$V_d = \left\{ \hat{\alpha} \in \mathbb{R}^n \middle| \|\hat{\alpha}\| = \|\alpha^*\|, \sum_{i=1}^{n} \hat{\alpha}_i \cdot K\big(x_i, x_j\big) > 0 \ \forall j \right\} \tag{20}$$

where n denotes the total number of training samples x_j and $\hat{\alpha}$ is an n-entry vector. The α^* corresponds to the output of the SVM training, constraining the version space to a hypersphere again.

Given a distinct form of discriminant functions, the cardinality of V_d is different from V. However, the optimal member in both version spaces that corresponds to the solution of the SVM should be the same due to the duality.

Similarly, for the dual problem of the soft margin SVM, if we follow the same adjustment performed in (18), i.e., using a soft margin SVM with given cost factor C to determine the relaxation ξ_i on all the training data x_i, the proposed soft margin version space can be redefined as:

$$V_d = \left\{ \hat{\alpha} \in [-C, C]^n \middle| \|\hat{\alpha}\| = \|\alpha^*\|, \sum_{i=1}^{n} \hat{\alpha}_i \cdot K\big(x_i, x_j\big) > -\xi_j \ \forall j \right\}. \tag{21}$$

Compared to the hard margin version space, the soft margin one is constrained within a smaller space $[-C, C]^n$ instead of the whole n-dimensional space $\mathbb{R}^n$. If $C = \infty$, all the ξ_j will be zero and thus the soft margin version space should be equivalent to the hard margin one.

B. Proposed Accelerated Active Learning With Probabilistic Version Space

Suppose that there is a pool of unlabeled data with a size of n, V_n is the resulting version space if we query all the instances in the pool and $\hat{w}_n^*$ is the corresponding optimal separating hyperplane, then $\hat{w}_n^*$ lies in version space V_i after i queries.

In case that the pool is of a huge size, it is too expensive to query the labels for all the instances in the pool. Since $\hat{w}_n^*$ is determined only by support vectors, a limited number of queries that include all the support vectors will also produce the same optimal hyperplane $\hat{w}_n^*$. Thus, $\hat{w}_n^*$ can be approximated with much lower expense.

Version space-based active learning tends to find a smart way of querying and finally has an accurate approximation of $\hat{w}_n^*$ with limited number of queries. For hard margin SVM, according to the definition of version space, $V_i \supseteq V_{i+1}$ for $i = 0, 1, 2, \ldots$, which means the size of the version space is reduced as more and more instances are queried. For soft margin SVM, as every query adds a cutting hyperplane in W, the version space would be reduced by more queries as well. If the size of V_k is small enough, any $\hat{w} \in V_k$ is close to $\hat{w}_n^*$ and we can use $\hat{w}_k^*$ to approximate the optimal classifier of the whole pool. The upper error bound of this approximation will be smaller if V_k is shrinking. Therefore, to make $\hat{w}_k^*$ converge to $\hat{w}_n^*$ as fast as possible, we can select query that shrink the version space as much as possible in every step.

Let Area(V_k) denote the surface area of the version space V_k which is defined by k queried samples. Since the shape of the version space could be very complicate in high-dimensional space, it is impractical to compute Area(V_k) explicitly. In general, $\hat{w}_k^*$ may be located near the center of the version space, thus the distance between $\hat{w}_k^*$ and support vectors, which corresponds to the margin of the unnormalized w_k^* in feature space, can be used to represent the surface area of the version space. We assume that Area(V_k) is proportional to the margin of w_k^* and, for simplicity, the constant of proportionality is 1

$$\text{Area}(V) = \frac{1}{\|w_k^*\|}. \tag{22}$$

Tong and Koller [13] tried to have the version space with every query as far as possible. They select such an instance in the pool in every step that has equal or most approximate size of version space no matter which class it is labeled. This method requires twice retraining of SVM for every instance in the pool, which is very expensive when it comes to cases that the pool is often of a huge size. And the reduction of version space in every step is likely to be not more than one half.

In this paper, we propose an accelerated active learning method with probabilistic version space. It reduces the retraining cost by pool size shrinking and tends to have a more aggressive reduction in version space.

In active learning process, version space described in the previous section is used to measure the resulting benefit of any candidate query. To compute the resulting reduction in version space of any candidate instance, we need to get its label and retrain an SVM by adding it into the training data set. We avoid querying the real label of the candidate instance by assuming its label as $+1$ and -1 and perform SVM retraining, respectively (no simulation is committed yet at this point). In every single step of active learning, if the pool has n instances, we should train $2n$ SVMs to find the optimal query. It is too expensive especially when it comes to our applications of circuit performance classifications.

Since most circuit parameters are of analog value, the pool consists of infinite unlabeled instances, which makes it impossible to calculate the expected size of resulting version space for every instance. One simple way is to randomly sample a certain number of instances in the input space to form a pool of finite size in every step. However, for a high-dimensional input space, it requires a huge number of samples for the algorithm to reach certain accuracy. It would be far too expensive to find an optimal query out of the pool.

In the SVM model, based on the definition of the constraints (4) and (12), for samples x_i right on the margin (i.e., support vectors in the hard margin SVM or support vectors with $\xi_i = 0$), we have $|f(x_i)| = 1$. According to (17), a new query of x will reduce the size of version space if and only if

$$|f(x)| < 1 \tag{23}$$

which means x should be closer to the hyperplane than support vectors with zero ξ_i. Therefore, in every step of active learning, we can sample a certain number of instances that satisfy (23) to form a pool with acceptable size instead of sampling in the whole input space and then perform previously mentioned active learning algorithm after that. Whichever instance in the pool is chosen to be queried, part of the version space will be cutoff, and hence, we can guarantee that the version space is reduced in every step.

In addition to the above acceleration, in every step of active learning, we try to find such an instance x in the pool that has the largest expectation of size reduction in version space. In other words, after we add x into the training data set, the expectation of the resulting version space should be the smallest. Let V_i denotes the version space after i queries and x_{i+1} denotes the $(i+1)$th query, define

$$V_i^+ = V_i \cap \left\{ \hat{w} \in W | + \hat{w} \cdot \phi(x_{i+1}) > -\frac{\xi_i}{\|w\|} \right\} \tag{24}$$

$$V_i^- = V_i \cap \left\{ \hat{w} \in W | - \hat{w} \cdot \phi(x_{i+1}) > -\frac{\xi_i}{\|w\|} \right\}. \tag{25}$$

Obviously, V_i^+ and V_i^- denote the version spaces that x_{i+1} is labeled as $+1$ or -1, respectively. A more aggressive strategy is to find an instance x_{i+1} in the $(i+1)$th step with smallest value of

$$E = P(y = 1|x_{i+1})\text{Area}\left(V_i^+\right) + P(y = -1|x_{i+1})\text{Area}\left(V_i^-\right) \tag{26}$$

where $P(y = 1|x_{i+1})$ and $P(y = -1|x_{i+1})$ denote the probability of x_{i+1} being labeled as $+1$ and -1, respectively.

We use an intuitive function for the conversion between SVM output $f(x)$ and the probability in our algorithm

$$P(y = 1|x) = \frac{1}{1 + \exp(-\alpha f(x))}. \tag{27}$$

Algorithm 1 Accelerated Active Learning With Probabilistic Version Space

1. Find an initial training data set S which includes both positive and negative examples
2. Train a soft margin SVM classifier $f(x)$ with S
3. Randomly pick up a huge set T from the input space
4. Classify T with $f(x)$
5. Pick up a certain number of instances with small enough $|f(x)|$ in T, form a subset $\hat{T}$
6. For every $x_i \in \hat{T}$,
   (a) Assign $P(y = 1|x_i)$ and $P(y = -1|x_i)$ to x_i
   (b) Label x_i as positive, add it to S temporarily
   (c) Train an SVM to compute its resulting size of version space $Area_+$
   (d) Label x_i as negative, add it to S temporarily, and train another SVM to get $Area_-$
   (e) $E = P(y = 1|x_i)Area_+ + P(y = -1|x_i)Area_-$
7. Query the real label of the instance with smallest E, add it into S
8. Repeat steps 2–7 until E is small enough
9. Train a final SVM with S

With a tuned parameter α, it provides symmetric probability results for computing the expectation of version space.

The algorithm flow of active learning combined by all the above steps is demonstrated in Algorithm 1.

It should be aware that the pool is randomly regenerated in each iteration, making it unique to the others in the other iterations. As the process iterates, the granularity of the superposition of all the pools in the previous iterations, which can be viewed as the discretization of the continuous input space, gets finer and finer. This can be used to capture the continuity of analog systems and makes the proposed method distinguished from the traditional active learning algorithms (such as [13]–[15]) for finite discrete applications.

C. Error Bound

An error bound estimation of the SVM classifiers based on leave-one-out cross validation is proposed in [2], providing a method to approximate the upper bound of the misclassification from a given training data set. However, such method is based on the assumption that all the training samples are selected independently. In the context of active learning, such assumption is always not true since new samples are always evaluated and selected by the current SVM classifier trained from the current training samples.

As mentioned in the previous section, standard SVM models with sufficient training data often produce good classifiers with promising performance. Therefore, rather than analyze the upper bound of the misclassification of the actively learned model, we treat the SVM classifier learned from sufficient data as the desired "golden" classifier and explore the maximum difference between such classifier and any other solutions in the reduced version space.

Due to the duality between the primal and dual forms of the SVM model, the optimal solution $\hat{w}^* \in V$ is equivalent to the optimal solution $\hat{\alpha}^* \in V_d$. Although V and V_d might have different cardinality, considering the powerful expressive capability of kernel method with (10), we make the assumption that $\hat{w} = \sum_{i=1}^{n} \hat{\alpha}_i \phi(x_i)$ for arbitrary $\hat{w}$, which guarantees the trends or convergence of applying active learning version space reduction in both version space definitions should be consistent.

As a result, the error between the golden classifier represented by $\hat{\alpha}^*$ and any solution $\hat{\alpha}$ in the version space can be formulated as

$$\Delta(\hat{\alpha}) = \left| f_d^{\hat{\alpha}^*}(x) - f_d^{\hat{\alpha}}(x) \right| = \sum_{i=1}^{n} \left| \hat{\alpha}_i^* - \hat{\alpha}_i \right| K(x_i, x). \quad (28)$$

If the Gaussian kernel (10) is applied here, since $0 \leq K(x, y) \leq 1$, we have $\Delta(\hat{\alpha}) \leq \sum_{i=1}^{n} |\hat{\alpha}_i^* - \hat{\alpha}_i|$. Taking all the training data into account, the upper bound of the difference between the golden classifier and any solution in the version space is the optimal solution of the following linear programming problem:

$$\max_{\hat{\alpha}} \ \sum_{i=1}^{n} \left| \hat{\alpha}_i^* - \hat{\alpha}_i \right| \quad (29)$$

$$\text{subject to} \ \sum_{i=1}^{n} \hat{\alpha}_i K\left(x_i, x_j\right) > 0 \ \forall j = 1, 2, \ldots, n \quad (30)$$

where x_i and x_j denote arbitrary training samples.

The procedure of active learning in the context of the error bound defined above is not as explicit as version space reduction. The reason is that adding a new sample x_{x+1} into the training data set might reduce the current set of the feasible solutions defined by (30), but a new variable $\hat{\alpha}_{n+1}$ will also be introduced. Nevertheless, since it suffices to consider only the feasible solutions on the boundary to find out the optimal solution (i.e., the error bound) of a linear programming problem, a new sample may help to lower the error bound if it cuts off a part of the boundary of the set of feasible solutions.

IV. Asymmetric Active Learning

Compared to related active learning methods developed in the machine learning domain [13]–[15], the proposed method mainly focuses on continuous problems rather than finite discrete space. In traditional discrete applications, the imbalance between the number of positive and negative samples is usually a challenging problem. While in circuit scenarios, since the continuous space can be infinitely discretized, meaning infinite number of different types of samples can be obtained, such problem may not be a concern any more. However, the need of safe prediction occurs as a new challenge in the circuit domain.

For example, in applications like circuit verification, we are much more concerned about the class of failure than the class of success. Misclassifying acceptable circuits into the class of failure may be allowed, while misclassifying flawed circuits into the class of success should be strictly restricted. In this light, we expect that the prediction on the failure class is conservative while the classifier still has high global accuracy.

Some adaptive sampling techniques such as [20]–[22] are developed to generate balanced training set from imbalanced data in the machine learning domain, and similar algorithms [9], [10] designed for circuit problems are adopted in the EDA community to solve the sampling problem of rare events (e.g., SRAM yield analysis). In this paper, the objective problem is different. Instead of dealing with imbalanced data density, we adjust our active learning algorithm to meet different safety requirements on the two classes.

A. Cost Asymmetric SVM

To reduce misclassifications in one class, one frequently used method is to assign a higher cost or penalty coefficient to that class [23], [24]. The previous soft margin SVM model will be further modified into the following cost asymmetric SVM model (also known as 2C-SVM):

$$\min_{w,\xi} \ \frac{\|w\|}{2} + C_+ \sum_{i:y_i=+1}^{n} \xi_i + C_- \sum_{i:y_i=-1}^{n} \xi_i \tag{31}$$

$$\text{subject to } \ y_i(w \cdot \phi(x_i)) \geq 1 - \xi_i, \xi_i \geq 0. \tag{32}$$

Without any loss of generality, assume that $C_+ > C_-$. As a result, rather than assigning a nonzero slack variable to a positive training sample, the 2C-SVM model tends to increase the slackness of one or more negative samples. Therefore, after the training process of 2C-SVM, fewer relaxation will be assigned to the positive training samples compared to the negative samples. The resulting classifier should be more accurate for positive instances.

The dual problem of the 2C-SVM shares the same objective function (6) while the constraints will be modified as

$$0 \leq \alpha_i \leq \begin{cases} C_+, & y_i > 0 \\ C_-, & y_i < 0. \end{cases} \tag{33}$$

By defining the class of failure as positive and the class of success as negative, if $C_+ \gg C_-$, then errors that misclassifying a positive instance as a negative one, also known as false negative, will be much fewer than misclassifications on negative instances, also known as false positive.

B. Safety Level Evaluation

It is hard to evaluate the safety level of a classifier on positive class explicitly, but apparently, a classifier with few false negatives and large margin on positive class is more conservative than those with more false negatives or smaller margin on positive class.

Hinge loss [25] is the loss function for soft margin SVM. For a training example (x, y), it is defined as

$$l(x, y) = \max(0, 1 - y \cdot f(x)). \tag{34}$$

It is widely used in SVM to penalize if (x, y) violates the margin, which is already implicitly included in the standard soft margin SVM and 2-C SVM. Recalling the definition of soft margin SVM, $l(x, y)$ is in fact ξ_i in (11) and (12). If $0 < l(x, y) \leq 1$, then (x, y) is the instance with distance to the hyperplane smaller than the margin. If $l(x, y) > 1$, then (x, y) is a misclassification.

In order to evaluate the conservation of the positive class, we sum up all the hinge loss in the positive class only

$$L = \sum_{y_i=+1} l(x, y). \tag{35}$$

The smaller L is, the more conservative the classifier is. If $L = 0$, since there is no misclassification or margin violation, we believe that it is the most conservative case.

C. Asymmetric Active Learning and Performance Optimization

To get a conservative classifier, we employ the 2C-SVM and the newly-defined hinge loss metric in the previous strategy to achieve a conservative active learning algorithm. In every step, we query an instance that may reduce the version space largely and maintain a small L at the same time. For every instance in the pool, we compute its expected loss as

$$\text{Loss} = P(y = 1|x_{i+1})L^+ + P(y = -1|x_{i+1})L^- \tag{36}$$

where L^+ and L^- denote the resulting hinge loss on the positive class if the instance is labeled as positive or negative, respectively. Then, we perform the following calculation for every instance:

$$D = E + \eta \cdot \text{Loss} \tag{37}$$

where η is the tradeoff coefficient and the instance with smallest D will be selected to be queried. This strategy may not shrink the version space the fastest, but it can make the classifier more conservative.

Since instance with smallest D is less likely to have a smallest E at the same time, the version space reduction in every step will be smaller and hence the convergence will be slower. In order to improve the efficiency, we propose the following global strategy for improving the convergence. Considering two 2C-SVMs with different cost coefficient but trained by the same training data set

$$\min_{w,\xi} \frac{\|w\|}{2} + C_{1+} \sum_{i:y_i=+1}^{n} \xi_i + C_{1-} \sum_{i:y_i=-1}^{n} \xi_i \tag{38}$$

$$\min_{w,\xi} \frac{\|w\|}{2} + C_{2+} \sum_{i:y_i=+1}^{n} \xi_i + C_{2-} \sum_{i:y_i=-1}^{n} \xi_i. \tag{39}$$

We know that $\|w\|$ and $\sum^n \xi_i$ are conflict notions that represent the margin and violation of the margin, respectively. Thus, $\|w\|$ will increase if $\sum^n \xi_i$ decreases and vice versa. For the two 2C-SVMs, if $C_{1\pm} > C_{2\pm}$, respectively, then we will have $\|w_1\| \geq \|w_2\|$. According to (22), it means that the size of the resulting version space of (38) will be smaller than that of (39). Assumes that C_+^* and C_-^* are the two cost coefficients that could achieve a conservative classifier in the final step, we may start the active learning from larger cost coefficients C_+ and C_- and gradually reduce them to C_+^* and C_-^*, respectively. By this strategy, we can achieve a faster convergence than the strategy that use C_+^* and C_-^* from the very beginning.

The algorithm flow of the asymmetric active learning is shown in Algorithm 2.

D. Dynamic Model Selection

According to the earlier discussions, the selection of the SVM model or, more specifically, the selection of the cost parameters (i.e., the C in the soft margin SVM and the C_+ and C_- in the 2C-SVM) plays an important role in the active learning procedure.

A popular choice of the algorithms on model selection is based on cross validation. Models with different parameters are trained and tested by cross validation and the one with the best average performance will be picked. However, in active learning, the number of training samples is limited and, again, samples are not independent with each other. Therefore, methods based on cross validation are not applicable in the procedure of active learning.

A simple yet practical analytic method for model selection is proposed in [26] to determine the appropriate parameters for the SVM regression. We adopt the similar reasoning here to determine the model parameter in each iteration during the active learning procedure.

Due to the fact that $0 \leq K(x, y) \leq 1$ for Gaussian kernel (10) and the dual form definition of the soft margin SVM, we have the following inequality:

$$\begin{aligned} |f(x)| &\leq \sum_{i=1}^{n_{\mathrm{SV}}} |\alpha_i y_i K(x_i, x)| \\ &\leq \sum_{i=1}^{n_{\mathrm{SV}}} |\alpha_i| \\ &\leq n_{\mathrm{SV}} \cdot C \end{aligned} \tag{40}$$

where x_i denote the support vectors (i.e., those with nonzero α_i) and n_{SV} denotes the total number of them. Then, the model parameter C can be estimated by

$$C \geq k \cdot \frac{\max_x |f(x)|}{n_{\mathrm{SV}}} \tag{41}$$

where k is a parameter used to implement the adjustment in C mentioned in the previous section. At the beginning of the active learning procedure, we use a large k for fast convergence, and then reduce it gradually toward 1 to get a better regularized solution. Since the input space x is often bounded in the circuit application, calculating $\max_x |f(x)|$ should be trivial once we obtain the form of $f(x)$.

In the application of active learning, the given initial training data set is often very small, and often easy to be completely separable. Thus, applying an SVM model with a large C will produce us the first approximation of the discriminant function $f(x)$ and the first group of support vectors. After that, in the nth ($n \geq 2$) iteration, we use the $f(x)$ and n_{SV} gained from the $(n-1)$th iteration to estimate the parameter C.

Another important model parameter is the ratio of C_+/C_- in the asymmetric model. One way to determine such ratio is to use the practical cost for the misclassification on different classes. However, such ratio might be indefinite or indistinct.

Algorithm 2 Asymmetric Active Learning

1. Find an initial training data set S which includes both positive and negative examples
2. Train a soft margin SVM classifier $f_1(x)$ with S
3. Randomly pick up a huge set T from the input space
4. Classify T with $f_1(x)$
5. Pick up a certain number of instances with small enough $|f_1(x)|$ in T, form a subset $\hat{T}$
6. For every $x_i \in \hat{T}$,
   (a) Assign $P(y = 1|x_i)$ and $P(y = -1|x_i)$ to x_i
   (b) Label x_i as positive, add it to S temporarily
   (c) Train a 2C-SVM with parameter C_+ and C_- to compute its resulting size of version space $Area_+$ and loss L_+
   (d) Label x_i as negative, add it to S temporarily, and train another 2C-SVM with parameter C_+ and C_- to get $Area_-$ and loss L_-
   (e) $D = P(y = 1|x_i)(Area_+ + \alpha L_+) + P(y = -1|x_i)(Area_- + \alpha L_-)$
7. Query the real label of the instance with smallest D, add it into S
8. If C_+ and C_- are larger than the given C_+^* and C_-^*, reduce C_- and C_+ accordingly
9. Repeat steps 2–8 until D is small enough
10. Train a final 2C-SVM with S and the cost coefficients C_+^* and C_-^*

In such cases, we can determine the ratio from the context of the model.

According to the definition of the 2C-SVM, an outlier x_i in the positive class that violates the margin may have an impact of $C_+K(x_i, x)$ to the learned discriminant function, since its corresponding $\alpha_i = C_+$. If it is in the negative class, then the impact should be $C_-K(x_i, x)$. The essence of the 2C-SVM is to make the impact of the two kinds of outliers different. Therefore, to ensure that outliers in the positive class always have greater impacts on the discriminant function than the negative outliers, we can select the ratio that satisfies

$$\frac{C_+}{C_-} \geq \frac{\max_{x_i, x_j} K(x_i, x_j)}{\min_{x_i, x_j} K(x_i, x_j)} \tag{42}$$

where x_i and x_j are arbitrary vectors in the input space. Again, since the input space in circuit application is often bounded, the maximum and minimum of the kernel function on the input space should be computable.

V. Experimental Results

In this section, our proposed method is applied as a general flow to various circuit applications. Since the flow mainly focuses on simulation need reduction, for very high-dimensional system, the running of a single simulation can still be expensive. Due to the limited scope, we illustrate the effectiveness of the proposed method in four simplified circuit systems.

Our proposed active learning approach employed in the following experiments is implemented in C++ on a Linux server.

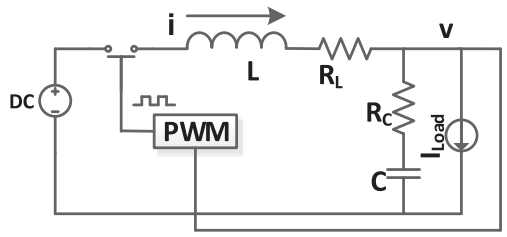


Fig. 4. Simplified model of an LC-based dc/dc converter.

Our active learning scheme interface with the base-line SVM package $\text{SVM}^{\text{Light}}$ [27]. And we use the default $\gamma = 1$ in this tool for SVM training. For the probability estimation in (27), the tuning parameter is set to be a constant $\alpha = 3$.

A. LC-Based DC/DC Converter Design Classification

A simplified model of an LC-based dc–dc converter is shown in Fig. 4, where R_L and R_C represent the parasitic resistance of the inductor L and capacitor C, respectively. The pulse width modulation (PWM) unit controls the power switch to keep charging or discharging the LC filter repeatedly, and hence generating both output voltage and inductor current ripples. Ripple noise is one of the important specifications of a dc/dc converter and, apparently, choices of design parameters L and C may produce different levels of ripple noise on the output voltage.

Assume that the switch, the voltage source, the PWM control module, and the feedback mechanism are all ideal components with perfect characteristics, and additionally the resistance of R_L and the conductance of R_C are proportional to the value of L and C, respectively. In this case, the state variables of the system can be defined as the current i on the inductor and the output voltage v if the input voltage V_{in} and the load current I_{Load} are given. And the dynamics of the system in the s-plane can be described by

$$v = \frac{(1 + sCR_C)[V_{\text{in}} - (sL + R_C)I_{\text{Load}}]}{1 + sC(R_C + R_L) + s^2LC} \tag{43}$$

and

$$i = \frac{sCV_{\text{in}} + (1 + sCR_C)I_{\text{Load}}}{1 + sC(R_C + R_L) + s^2}. \tag{44}$$

By assuming that both v and i are initially set to be zero, the simulation is implemented using numerical methods to approximate (43) and (44). In terms of efficiency, active learning is not superior compared to passive learning due to the inexpensive simulations. This 2-D system is used as an example to show the intuition of the convergence of active learning and the predominant improvement in reducing the need of samples/simulations.

The goal of our analysis here is to examine the dependencies of the ripple on important design parameters under a fixed working condition (i.e., with fixed input voltage and load current). For this, an SVM model can be extracted by defining the input features as L and C and labeling each instance according to its ripple noise. Given a threshold noise level, if a circuit with a group of design parameters has a ripple noise that exceeds the threshold, it will be labeled as a positive sample. Otherwise, the negative label meaning that it meets the given threshold will be assigned to it. In this experiment, we set the load current $I_{\text{Load}} = 0.1$ A, the input voltage as 1 V, and the threshold of ripple noise as 0.2 mV.

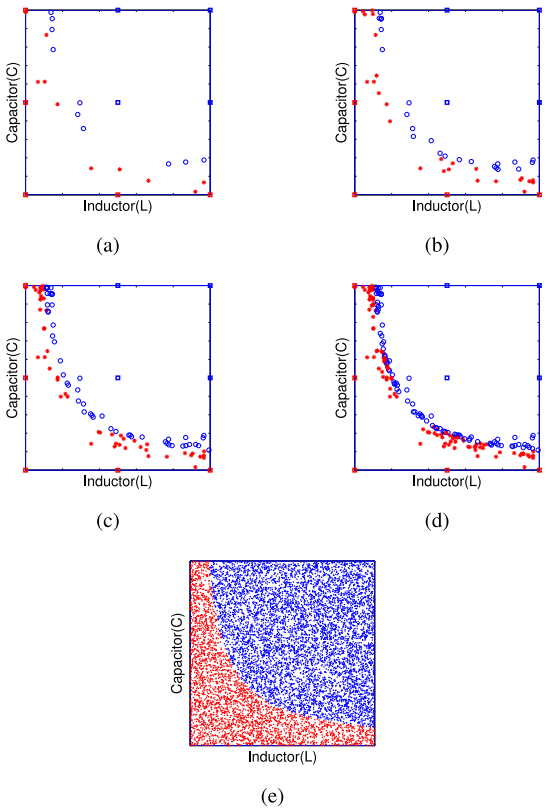


Fig. 5. Sampling procedure of active learning in the parameter space L : [1 nH, 1 uH] $\times$ C : [10 nF, 10 uF] (red star denotes positive instances, blue circle denotes negative instances, and square denotes initial training samples). (a) 20, (b) 50, (c) 100, and (d) 200 samples selected. (e) 10 000 random instances.

The objective of the classifier is to accurately predict whether a combination of L and C within a given area will meet the given ripple noise specification or not. Thus, the proposed symmetric model is adopted. We evenly select nine samples out of the input space as the initial training data set and the procedure of active learning is shown in Fig. 5. In the query process for an unknown pair of (L, C), a transient simulation of (43) and (44) is invoked and the ripple noise is measured after the system gets stable, i.e., the current on the inductor and the output voltage settle down.

From Fig. 5(a)–(d), the convergence of the active learning sampling is clearly demonstrated. New samples tend to be selected around the boundary between the positive and negative classes and gradually cluster the accurate separating hyperplane. The 200 samples selected by the active learning scheme in Fig. 5(d) already shows the outline of the hyperplane. Compared to Fig. 5(e) that shows the hyperplane with a large number of random instances, the approximated hyperplane in Fig. 5(d) is very close.

The learned classifier can be exploited to quickly determine the performance of a certain design. Compared to SVM training based on random sampling, the required number of

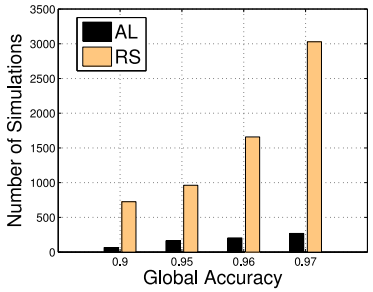


Fig. 6. Comparison in the required number of simulations (AL, active learning; RS, random sampling) where the global accuracy is defined as the fraction of the correct predictions on 10 K random test samples (e.g., 0.1 means 10%).

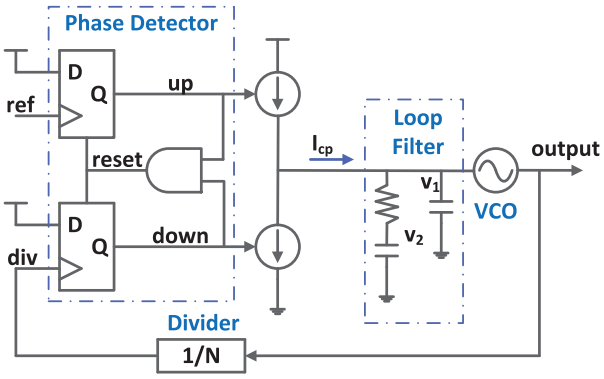


Fig. 7. Block diagram of PLL.

simulations is much fewer in active learning. Such comparison with respect to the number of simulations is shown in Fig. 6 and a large reduction in simulations is achieved by active learning. For example, active learning only needs 10% of the total number of simulations of the random sampling to produce a classifier with 97% accuracy.

In general, the active learning flow can be extended to other circuit design problems. We believe building an accurate classifier for circuits by taking various design parameters as its input features is meaningful for quick performance prediction in circuit design/debug. Once the classifier is given, designers can directly tell whether a design with some new parameters is going to work or the training cost to obtain such classifier can be significantly reduced by our proposed active learning scheme.

B. PLL Verification

The charge pump PLL being verified is shown in Fig. 7. If the divided output signal is ahead of the reference clock, the voltage on the node down will be logical 1 and up be logical 0, controlling the charge pump to discharge the two capacitors in the loop filter to lower the input voltage of voltage-controlled oscillator (VCO), which will lower the frequency and reduce the phase of the output signal, or vice versa. If the phase difference between the reference clock and divided output signal is 0, both up and down are logical 0. Through such negative feedback, the frequency of the output voltage should gradually converge and finally be locked to N times the reference frequency.

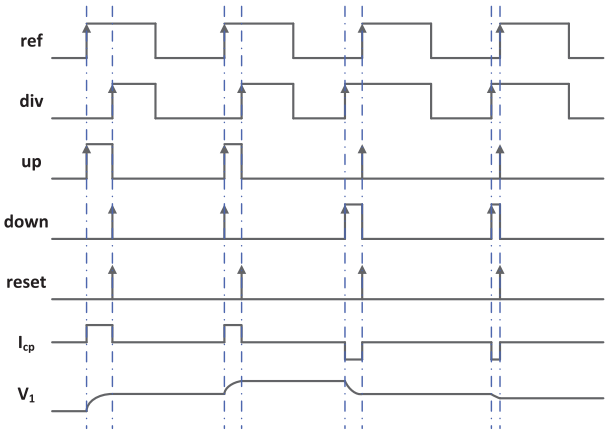


Fig. 8. Timing diagram of the PD and the resulting output of the charge pump and loop filter.

Typically, the phase detector (PD) is implemented by two D flip-flops and one AND gate. The behavior of the PD can be described as: if the divided signal div takes the lead, a rising edge will be generated at down, followed by a falling edge triggered by the arrival of the rising edge of the reference clock ref. A similar pulse will be generated at up if reference clock ref takes the lead. A pulse at up makes the charge pump to charge the loop filter and increase the input voltage of the VCO, while a pulse at down reverses such impact. The timing diagram of the PD and the resulting output of the charge pump and loop filter is shown in Fig. 8.

We use Verilog-A to set up the behaviorial model of the PLL and, similar with the PLL model built in [28], define the state of the circuit with five variables: 1) voltages on two capacitors v_1 and v_2; 2) phase difference ϕ; and 3) two digital variables up and down.

Our goal is to verify that PLL with various size of initial state space will be locked within a certain time or not. To start with some failures for the SVM training, we set a rather large searching space for the state variables that is very likely to break the PLL (which is a similar concept in stress-testing). The searching space for the two initial voltages is the interval [0, 0.8 V] and the phase difference has a range $[0, 2\pi)$. There are only three combinations of up and down: {0, 0}, {0, 1}, {1, 0}.

In this case, we hope to have a conservative verification of PLL lock-time over uncertain initial startup conditions. Thus the asymmetric active learning algorithm is employed. The class with positive label corresponds to meeting the specified lock time, and the class with negative label corresponds to failing the lock-time specification. We compare the performance of this method with another method that only trains a 2C-SVM with random samples to illustrate the improvement in efficiency of the our method.

We use a set of 27 instances evenly distributed in the whole state space as our initial training data set. To make the comparison more reasonable, we also add this set into the random samples of the uniform sampling. The results of classifying a randomly generated data set with a size of 100 000 by two methods are shown in Fig. 9. We can infer that much less simulations and runtime is needed for our active learning method

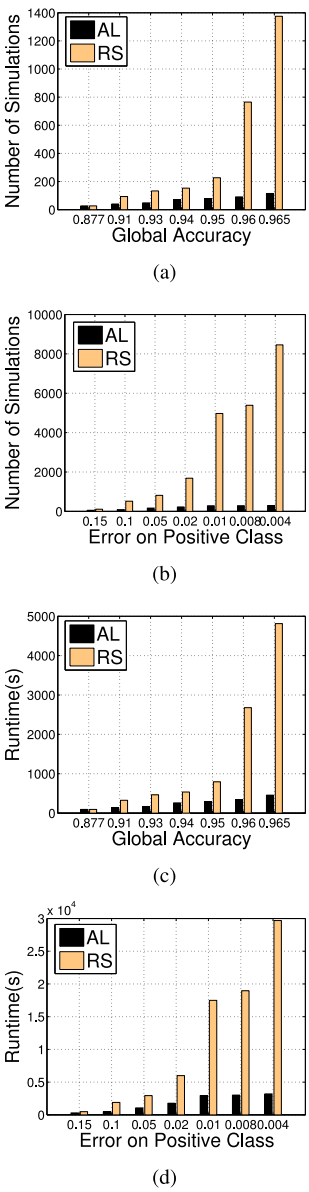


Fig. 9. Results of PLL experiments (AL, active learning; RS, random sampling) where the global accuracy and the error on positive class are defined as the fraction of the correct predictions, with a test set whose size is 100 K (e.g., 0.1 means 10%). Number of simulations for (a) certain accuracy and (b) safe prediction. Runtime to achieve (c) certain accuracy and safe prediction.

to achieve certain accuracy and prediction safety levels, the latter of which is measured by the classification error on the positive class (i.e., percentage of false negatives over the positive class). Consider achieving same accuracy, the number of simulations is reduced by an average of 70% and a maximum of 92%, the runtime is reduced by an average of 69% and a maximum of 91%. Considering the safety, an average of 89% and a maximum of 96% simulations are saved, with an average of 72% and a maximum of 89% saving in runtime.

By neglecting the two digital state variables, the distributions of instances sampled by active learning is shown in Fig. 10(a). Again, the result shows that samples selected by active learning tend to cluster the separating hyperplane. A further projection into a 2-D space (V_1, V_2) clearly shows

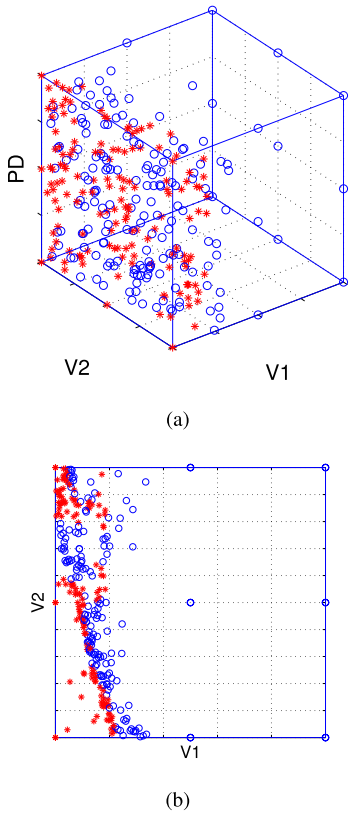


Fig. 10. Samples selected by active learning. (a) Projection of the selected samples in the continuous 3-D $(V_1, V_2, \Delta\phi)$ space, where V_1 and V_2 are the initial voltages on the two capacitors, and $\Delta\phi$ is the initial phase difference. (b) Projection of the selected samples in the (V_1, V_2) space.

the trend. The projection also implies that the initial states V_1 and V_2 have more impacts on the lock time of the PLL.

Note that the verification problem defined in this experiment is different from what is commonly approached as the problem of formal verification, such as [28] and [29] based on reachability analysis and [30] and [31] based on global convergence analysis. Their identical concern is to explore the problematic initial states in a fixed initial state space. However, in our experiment, we aim at determining proper initial state spaces for a design. Once an SVM classifier is trained for a given PLL, if the actual initial state space is a subset of the class of success, its confident to accept the design. If not, certain suggestions can be fed back to designers to fix the problem. For example, precharging mechanism before or during the power-on process can be added to the PLL to shift the original state space to the desired space. The model and the flow are also reusable if the design is changed in the events like redesigning the circuit, or embedding the circuit into a larger system, and so on.

C. Ring Oscillator Reliability Analysis

As the third case study, we analyze the reliability for the three-inverter ring oscillator with respect to its process variations and initial conditions. The experiment follows similar methodology of yield analysis. We generalize the problem to circuit reliability analysis since the system also takes initial conditions into account in addition to process variations.

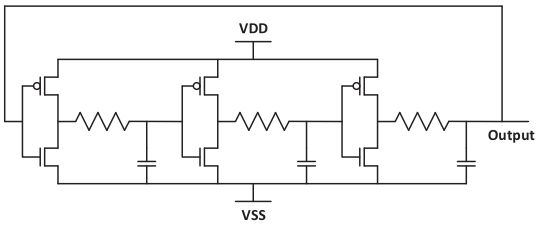


Fig. 11. Transistor level schematic of the three-inverter ring oscillator.

We use the proposed active learning scheme to train accurate symmetric SVM classifiers (since the accuracy is often more important than the conservativeness in reliability analysis) and contrast with the standard SVM generated by uniform sampling. While it is possible to combine our SVM classifiers with a statistical sampling technique as in [8], due to the scope limitation, we only demonstrate the improved performance of our active-learning-based SVM approach.

The transistor level circuit for the ring oscillator is shown in Fig. 11. Performances of the ring oscillator are affected by both process variation of the six transistors and initial conditions, i.e., the initial voltages on the three capacitors, of the ring oscillator. For example, the settling time and the output frequency of the oscillator is determined by each stage, which involves the delay of the inverter (affected by process variations) and the settling time of the RC filter (affected by the initial conditions). Furthermore, the output swing is also vulnerable to process variations due to the process variation sensitivity of the ON-state effective resistance and the OFF-state current of the transistors.

In this experiment, we use 1 kΩ as the value of the resistors and 10 pF as the value of the capacitors. We define that a ring oscillator is reliable if its output cycle is smaller than 110 ns and its output swing is greater than 0.7 V when we set V_{DD} as 1 V. The reliability of the ring oscillator is the probability of being reliable with respect to the given distributions of process variations and initial conditions.

The employed transistor model is the Berkeley short-channel IGFET model 4 model of IBMs 90 nm technology. For every transistor in Fig. 11, its modeled process variations include the gate oxide thickness t_{ox}, the channel length fluctuation ΔL, and the threshold voltage V_{th0}, each following a Gaussian distribution with $\sigma = 14\%$ ($\pm 42\%$ for 3σ). For the initial voltages on the three capacitors, they are modeled as uniform random variables in the interval $[V_{\mathrm{SS}}, V_{\mathrm{DD}}]$. These parameters form a 21-dimensional input space. Symmetric SVM classifiers are trained with different sampling methods in this space to predict whether an input vector is reliable or not. Once an accurate classifier is learned, a large volume of input vectors that are randomly generated from the given distributions can be quickly classified, leading to an efficient approximation of the reliability.

At first, we run Monte Carlo simulations with 100 K samples (i.e., 100 K circuit simulations in total) and calculate the ratio of reliable instances. Such ratio is treated as the accurate reliability. Then, we figure out the minimum number of simulations and the minimum runtime needed in the two sampling schemes to converge to the Monte Carlo reliability $\pm 1\%$ with a fluctuation smaller than 3%. The active learning starts with an initial training set of 100 random instances and picks new samples with the proposed strategy, while uniform sampling just keeps adding random samples into the training data set. The results of the two method are shown in Table I. We also list the cost of the Monte Carlo simulations in the table to show the benefit of SVM. The minimum cost of the Monte Carlo method is defined as the moment when the fluctuation of the prediction is smaller than 3%.

TABLE I
COMPARISON OF DIFFERENT SAMPLING SCHEMES

	Monte Carlo	Uniform Sampling	Active Learning
Average Predicted Reliability After Convergence	81.91%	82.17%	81.54%
Min # of Simulations	32k	9.55k	3.42k
Min Runtime($\times 10^3$s)	86.4	25.8	18.1
Overhead of the Sampling Algorithm($\times 10^3$s)	0	0	8.9
Reduction in Simulations	-	70.2%	89.3%
Speedup in Runtime	-	3.35	4.77

Compared with Monte Carlo simulation, building an SVM classifier to approximate the reliability can significantly reduces cost in terms of the number of simulations and the runtime. And the proposed active learning scheme can further reduce the need of circuit simulations and improve the overall efficiency. Although there is overhead for the sampling algorithm, the active learning flow provides a promising speedup for the reliability analysis.

D. Prediction of Peak Chip Temperature Using Limited Number of On-Chip Thermal Sensors

In this experiment, finite elements simulations are employed to check whether the temperatures on a chip exceed the highest allowable value or not via limited number of sensors integrated on the chip. Fig. 12 shows the functional blocks placement of a processor, similar to the simplified Digital Equipment Corporation alpha chip model in [32]. We place five sensors on top five blocks with highest possible power density (the positions of the red squares in Fig. 12).

Full-chip thermal simulation is performed by running a finite-difference based in-house thermal simulator. The simulator adopts the conjugate gradient iterative method with an incomplete lower upper preconditioner. On a Linux-based desktop, each thermal simulation takes about 65 s to complete. This simulation runtime will grow if a finer thermal partial differential equation discretization is adopted or finer details of material inhomogeneity is considered along the lateral dimensions of each material layer on the chip, producing a higher demand for smart learning.

For every block, the parameter is the power consumption that varies from 0 to its peak value. The parameters of all the

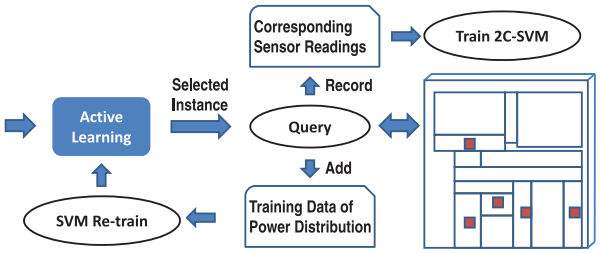


Fig. 12. Functional blocks placement and sensor positions and experiment flow.

15 blocks form a 15-dimensional space. Our goal is to train an accurate and conservative SVM classifier that can take the readings from the five sensors and then predict the actual peak chip temperature, which may not take place at any of these sensor locations. To guarantee the safety of the chip, we use asymmetric models in two steps.

In the first step of the flow, we train a 2C-SVM classifier that maps from block level power consumptions to the actual peak chip temperature. The classifier takes the power consumption on all the blocks as the input features, and the label/prediction is made according to the given temperature threshold. During this step, if any simulation is invoked to query the real label of a certain combination of the 15 parameters, the sensor readings are also recorded and paired with the obtained label. Active learning is invoked in this step to select samples with high quality.

In the second step, we use the recorded sensor readings and their corresponding labels to train another 2C-SVM classifier which makes the prediction of peak temperature directly based on sensor readings. This step assembles the fine samples generated from the last step into a classifier with desired format.

We use a set of 100 random instances as the initial training data set for the active learning. To provide a comparison reference, we repeat the above two-step process based on passive learning, i.e., training on a large set of random samples. Again, we add this set into the random samples of the passive learning. The results of classifying a randomly generated data set with a size of 100 000 by two methods are shown in Fig. 13. The active learning costs significantly less in terms of simulation and runtime. In terms of accuracy, we save an average of 51% and up to 67% in simulations as well as an average of 38% and up to 54% in runtime. In terms of achieving the same level of safety guarantee (e.g., false negatives), there is an average of 71% and up to 84% reduction in the needed number of simulations together with an average of 63% and up to 87% reductions in runtime.

This experiment shows the potential application of the proposed flow in monitoring complex systems. While implementing a substantial number of sensors in the system is beneficial to the safety, it may be costly for complex and highly integrated systems. In this light, the active learning guided sampling and SVM techniques can be applied to reduce the required number of sensors while retaining high alertness in the observation of the working conditions of the systems.

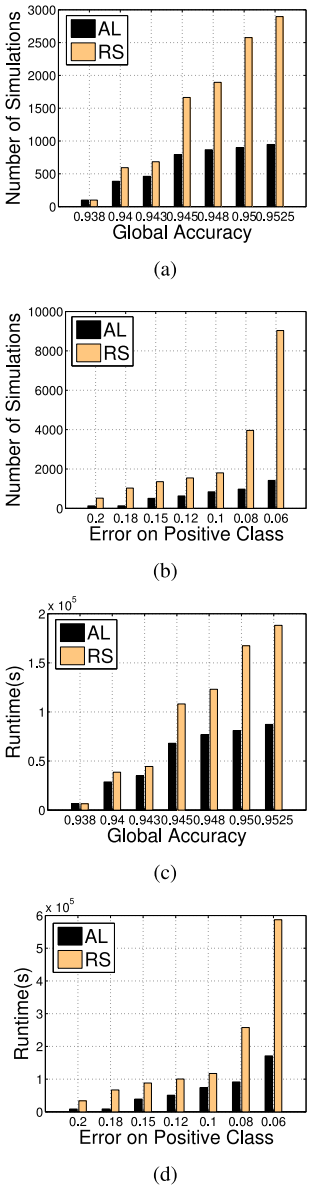


Fig. 13. Results of chip thermal experiments (AL, active learning; RS, random sampling) where the global accuracy and the error on positive class are defined as the fractions of the correct predictions, using a test data set whose size is 100 K (e.g., 0.1 means 10%). Number of simulations for (a) certain accuracy and (b) safe prediction. Runtime to achieve (c) certain accuracy and (d) safe prediction.

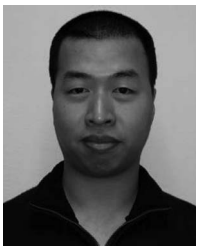
VI. Conclusion

In this paper, we presented an SVM-based active learning method for circuit performance classification. We built up two algorithm flows: 1) the symmetric one is for classifications only considering the accuracy and 2) the asymmetric one concerns about both safety and accuracy. We applied the proposed method in dc/dc converter ripple noise analysis, PLL lock-time verification, ring oscillator reliability analysis and chip thermal checking, and provide significant efficiency improvement in all the experiments.

References

[1] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[2] V. N. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.
[3] H. Liu, A. Singhee, R. A. Rutenbar, and L. R. Carley, "Remembrance of circuits past: Macromodeling by data mining in large analog design spaces," in *Proc. 39th IEEE/ACM Design Autom. Conf. (DAC)*, New Orleans, LA, USA, 2002, pp. 437–442.
[4] F. De Bernardinis, M. I. Jordan, and A. Sangiovanni-Vincentelli, "Support vector machines for analog circuit performance representation," in *Proc. IEEE/ACM Design Autom. Conf. (DAC)*, Anaheim, CA, USA, Jun. 2003, pp. 964–969.
[5] D. Drmanac, B. Bolin, L.-C. Wang, and M. S. Abadir, "Minimizing outlier delay test cost in the presence of systematic variability," in *Proc. IEEE Int. Test Conf. (ITC)*, Austin, TX, USA, Nov. 2009, pp. 1–10.
[6] E. Maricau, D. De Jonghe, and G. Gielen, "Hierarchical analog circuit reliability analysis using multivariate nonlinear regression and active learning sample selection," in *Proc. IEEE Design Autom. Test Europe Conf. Exhibit. (DATE)*, Dresden, Germany, 2012, pp. 745–750.
[7] P. Bastani, N. Callegari, L.-C. Wang, and M. S. Abadir, "Statistical diagnosis of unmodeled systematic timing effects," in *Proc. 45th IEEE/ACM Design Autom. Conf. (DAC)*, Anaheim, CA, USA, Jun. 2008, pp. 355–360.
[8] A. Singhee and R. A. Rutenbar, "Statistical blockade: Very fast statistical simulation and modeling of rare circuit events and its application to memory design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 8, pp. 1176–1189, Aug. 2009.
[9] C. Dong and X. Li, "Efficient SRAM failure rate prediction via Gibbs sampling," in *Proc. 48th IEEE/ACM Design Autom. Conf. (DAC)*, New York, NY, USA, Jun. 2011, pp. 200–205.
[10] R. Kanj, R. Joshi, and S. Nassif, "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events," in *Proc. 43rd IEEE/ACM Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2006, pp. 69–72.
[11] J. Yao, Z. Ye, and Y. Wang, "Importance boundary sampling for SRAM yield analysis with multiple failure regions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 3, pp. 384–396, Mar. 2014.
[12] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *Proc. 11th Int. Conf. Mach. Learn.*, New Brunswick, NJ, USA, 1994, pp. 148–156.
[13] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, Mar. 2002.
[14] B. Settles, "Active learning," *Syn. Lect. Artif. Intell. Mach. Learn.*, vol. 6, no. 1, pp. 1–114, 2012.
[15] J. Kremer, K. S. Pedersen, and C. Igel, "Active learning with support vector machines," *Wiley Interdiscipl. Rev. Data Min. Knowl. Disc.*, vol. 4, no. 4, pp. 313–326, 2014.
[16] S.-J. Huang, R. Jin, and Z.-H. Zhou, "Active learning by querying informative and representative examples," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Vancouver, BC, Canada, 2010, pp. 892–900.
[17] E. Pasolli, F. Melgani, and Y. Bazi, "Support vector machine active learning through significance space construction," *IEEE Geosci. Remote Sens. Lett.*, vol. 8, no. 3, pp. 431–435, May 2011.
[18] A. Beygelzimer, J. Langford, Z. Tong, and D. J. Hsu, "Agnostic active learning without constraints," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2010, pp. 199–207.
[19] S. Ertekin, J. Huang, L. Bottou, and L. Giles, "Learning on the border: Active learning in imbalanced data classification," in *Proc. 16th ACM Conf. Conf. Inf. Knowl. Manage.*, Lisbon, Portugal, 2007, pp. 127–136.
[20] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN) (IEEE World Congr. Comput. Intell.)*, Hong Kong, 2008, pp. 1322–1328.
[21] S. Chen, H. He, and E. A. Garcia, "RAMOBoost: Ranked minority oversampling in boosting," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1624–1642, Oct. 2010.
[22] S. Wang, Z. Li, W. Chao, and Q. Cao, "Applying adaptive over-sampling technique based on data density and cost-sensitive SVM to imbalanced learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Brisbane, QLD, Australia, 2012, pp. 1–8.
[23] E. Osuna, R. Freund, and F. Girosi, "Support vector machines: Training and applications," Massachusetts Inst. Technol., Cambridge, MA, USA, AI Memo 1602, 1997.
[24] F. R. Bach, D. Heckerman, and E. Horvitz, "Considering cost asymmetry in learning classifiers," *J. Mach. Learn. Res.*, vol. 7, no. 2, pp. 1713–1741, 2007.
[25] G. Wahba *et al.* "Support vector machines, reproducing kernel Hilbert spaces and the randomized GACV," in *Advances in Kernel Methods-Support Vector Learning*, vol. 6. Cambridge, MA, USA: MIT Press, 1999, pp. 69–88.
[26] V. Cherkassky and Y. Ma, "Practical selection of SVM parameters and noise estimation for SVM regression," *Neural Netw.*, vol. 17, no. 1, pp. 113–126, 2004.
[27] T. Joachims, "Making large scale SVM learning practical," in *Advances in Kernel Methods*. Cambridge, MA, USA: MIT Press, 1999.
[28] L. Yin, Y. Deng, and P. Li, "Simulation-assisted formal verification of nonlinear mixed-signal circuits with Bayesian inference guidance," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 7, pp. 977–990, Jul. 2013.
[29] H. Lin and P. Li, "Parallel hierarchical reachability analysis for analog verification," in *Proc. 51st Annu. Design Autom. Conf.*, San Francisco, CA, USA, 2014, pp. 1–6.
[30] S. Youn, J. Kim, and M. Horowitz, "Global convergence analysis of mixed-signal systems," in *Proc. 48th Design Autom. Conf.*, New York, NY, USA, 2011, pp. 498–503.
[31] T. Kim *et al.*, "Verifying start-up failures in coupled ring oscillators in presence of variability using predictive global optimization," in *Proc. Int. Conf. Comput.-Aided Design*, San Jose, CA, USA, 2013, pp. 486–493.
[32] P. Li, L. T. Pileggi, M. Asheghi, and R. Chandra, "IC thermal simulation and modeling via efficient multigrid-based approaches," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 9, pp. 1763–1776, Sep. 2006.



Honghuang Lin received the B.S. degree in automation from Tsinghua University, Beijing, China, in 2011. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA.

His current research interests include analog and mixed-signal circuit verification, machine learning-based circuit analysis, statistical circuit modeling, and physiological signal processing.

Peng Li (S'02–M'04–SM'09) received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2003.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA. His current research interests include the general areas of circuits and systems, electronic design automation, and computational neuroscience.

Prof. Li was a recipient of several awards, including the IEEE/ACM William J. McCalla International Conference on Computer Aided Design Best Paper Award, the IEEE/ACM Design Automation Conference Best Paper Award thrice, the Semiconductor Research Corporation Inventor Recognition Award twice, the Microelectronics Advanced Research Corporation Inventor Recognition Award twice, and the National Science Foundation CAREER Award, and the Electrical and Computer Engineering Outstanding Professor Award from Texas A&M University and was named as a TEES Fellow and William O. and Montine P. Head Faculty Fellow. He was an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 2008 to 2013. He is also an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-II: EXPRESS BRIEFS. He has served on the committees of several international conferences and workshops.