A Probably Approximately Correct Framework to Estimate Performance Degradation in Embedded Systems

Cesare Alippi, *Senior Member, IEEE*

***Abstract*—Future design environments for embedded systems will require the development of sophisticated computer-aided design tools for compiling the high-level specifications of an application down to a final low-level language describing the embedded solution. This requires abstraction of technology-dependent aspects and requirements into behavioral entities. The paper takes a first step in this direction by introducing a high-level methodology for estimating the performance degradation of an *application* affected by perturbations; a special emphasis is given to accuracy performance. To grant generality it is uniquely assumed that the performance degradation function and the mathematical formulation describing the application are Lebesgue measurable. Perturbations affecting the application abstract details related to physical sources of uncertainties such as finite precision representation, faults, fluctuations of physical parameters, battery power variations, and aging effects whose impact on the computation can be treated within a high-level homogenous framework. A novel stochastic theory based on randomization is suggested to quantify the approximated nature of the perturbed environment. The outcomes are two algorithms which estimate in polynomial time the performance degradation of the application once affected by perturbations. Such information can then be exploited by HW/SW codesign methodologies to guide the subsequent partitioning between HW and SW, analog versus digital, fixed versus floating point, or used to validate architectural choices before any low-level design step takes place. The proposed method is finally applied to real designs involving neural and wavelet-based applications.**

***Index Terms*—Approximated computation, embedded system design, high-level design, randomized algorithms, sensitivity analysis.**

I. Introduction

TIME-TO-MARKET, cost, device size, and low-power requirements are pushing the research in embedded applications toward the development of sophisticated high-level design tools. The "dream" is to describe the application at a very high level and generate, by means of a compiler, a formal target description suitable for final implementation.

The first step to be taken in this direction requires moving the formalization of the project and its specifications toward the highest abstraction levels; the second step addresses the development of suitable computer-aided design (CAD) tools for compiling the application and its specifications to a synthesizable description.

Manuscript received April 20, 2001; revised February 9, 2002. This paper was recommended by Associate Editor R. Gupta.

The author is with the Dipartimento di Elettronica e Informazione, Politecnico di Milano, 20133 Milano, Italy (e-mail: Alippi@elet.polimi.it).

Publisher Item Identifier S 0278-0070(02)05627-0.

Unfortunately, present commercial design environments [1], despite their effectiveness, characterize the project specifications only at the RT-functional/behavioral levels by directly taking into account features such as time, cost, and, sometimes, power consumption. In some cases further details are needed if not the whole description of the HW/SW physical architecture [2], [3].

In this paper, we move in the abstraction direction by formalizing what we can loosely define as "performance degradation" of an application, namely a measure of adherence between the performance of the implemented computation, subject to architectural/physical constraints, and the ideal unconstrained one. The classic scenario is represented by a sophisticated signal/image processing embedded application where the physical implementation somehow limits the application performance because of cost, memory, device size, and power constraints [4].

General performance degradation estimates cannot be extracted by existing tools [1] which, in the best case, only measure the impact of a finite precision representation of the computational flow on accuracy for a given input. In such cases, the designer fixes the number of bits for the different architectural entities, selects the quantization operator (e.g., truncation, rounding, or jamming), the architecture supporting the computation and the input pattern. The simulator, by propagating the input, provides the actual error-affected output. Such simulators are limited to finite precision aspects, cannot support a mixed digital/analog analysis, and do not provide any confidence index for the obtained results. In addition, a change in the quantization operator implies a new simulation and the analysis provides only a local indication of the accuracy performance degradation in correspondence with the specific input.

Yet, effective and reliable application-level performance estimates can be exploited by lower level synthesis layers, e.g., by dimensioning the data word length, deciding if a floating point operations is needed or a fixed point operation can be used instead, even solving the issue of analog versus digital implementation for the different parts composing the computation. Such aspects solely depend on the robustness of the application and have in turn a great impact on cost, memory dimensioning, low-power consumption, and accuracy performance. For instance, if a module composing the computation is robust enough once affected by behavioral perturbations then high precision (e.g., floating point) is not necessary and a fixed point digital or analog solutions can be envisaged instead. Moreover,

a high robustness degree implies that a reduced number of bits can be considered to represent a module computation with a consequent positive impact on power consumption and memory size. The loss in performance–accuracy induced by perturbations must be measured with a suitable figure of merit designed to characterize the relevant features of the application. Such information will be used in subsequent lower design layers to guide the designer toward the identification, dimensioning and characterization of the final physical architecture, e.g., as done in [5] and [6] for linear or linearisable applications and a noise to signal ratio figure of merit. Therefore, determination of a reliable measure for the performance loss induced by behavioral perturbations can be effectively exploited in subsequent low-level design phases.

The paper addresses the performance degradation/perturbation issue at the very high level and, hence, independently from implementation or technological aspects or any other low-level feature. The performance–accuracy aspect can then be immediately extended to cover different problems related to the computer arena since computation accuracy can be seen as a case of a more general framework addressing the performance variation induced by some unspecified sources of perturbation. For instance, the suggested methodology can also be applied to analyze the impact of fluctuations in battery power on performance, where performance must be suitably defined as a function of clock frequency, throughput, latency, etc. In the following, the focus is on the loss in performance accuracy but the reader can suitably extend the theory to the specific performance degradation case.

The methodological high-level framework for accuracy estimation is based on the intuitive concepts of perturbations and perturbed computation.

Definitions:

1) The computational accuracy index is an estimate of the accuracy loss induced by perturbations affecting the application.
2) The reference computation is the computation associated with the algorithm solving the application.
3) The perturbed computation is the reference computation affected by perturbations injected in a set of perturbation injection points.
4) A perturbation in a point of the computational graph is a discrepancy between the ideal "error free" computation provided by the reference computation and that provided by the perturbed computation when the other sources of perturbations have been switched off.

Here, perturbations represent abstractions of physical uncertainties affecting the computation. The key point of the analysis is based on the fact that if a module is robust with respect to perturbations defined within a domain, then all physically related sources of uncertainty associated with the embedded system implementation or errors arising during its operational life and belonging to such domain will induce a tolerable performance loss.

Common sources of errors affecting the computation, which *de facto*, can be intended as particular realizations of the perturbation, are finite precision representations, deviation of parameters from nominal values due to the production process or aging effects, faults affecting the device, fluctuations in battery power or any other well-defined source of uncertainty. Therefore, fixed-point analysis addressed by present CAD tools is only a special case of a more general framework and, as such, will be here suitably extended and completed.

The novel formal methodology here suggested for estimating the performance–accuracy loss of an application acts directly at the application level and can be integrated in existing CAD environments, and it is general and removes the hypotheses assumed in the related sensitivity analysis literature to make the mathematics more amenable. Several authors focus attention on very specific and limited classes of computation [7]–[10], consider linearized analyses based on some small perturbation hypothesis [5]–[9], postulate particular properties, and assume unnatural behaviors for the perturbation/computation [6], [8]. Since our main aim is generality, we remove ALL limiting hypotheses. In particular, the suggested methodology:

- is applicable to all Lebesgue measurable applications;
- can be used with all Lebesgue measurable figures of merit;
- deals with generic stationary perturbations;
- is characterized by a polytime complexity in exploring the perturbation space to estimate the performance loss of the application and can be easily implemented within a CAD environment;
- is independent from low-level implementations and technological details;
- extends and completes finite precision analysis present in available CAD tools (which constitute a particular subcase of the theory).

The structure of the paper is as follows. Section II introduces the concept of probably approximately correct computation, the theoretical framework developed to characterize and estimate the accuracy of a perturbed computation. Section III provides the skeleton of the methodology by introducing the polytime algorithms derived from the probably approximately correct computation theory and shows how the methodology can be integrated within CAD tools. Experiments are finally given in Section IV where the theory is applied to two real applications.

II. A Probably Approximately Correct Computation

A perturbation affecting a computational flow transforms the error-free reference computation in the perturbed counterpart and, hence, the ideal computation in an approximation. We formalize in the following the key elements of the perturbation analysis.

Denote by $y = f(x)$, $y \in Y \subset \Re^k$, $x \in X \subseteq \Re^d$ the mathematical description of the reference computation and by $\Delta \in D \subseteq \Re^p$ a generic p-dimensional perturbation vector, a component for each independent perturbation affecting $f(x)$. Depending on the application, the characterization of the input space X and the perturbation space D, the analysis can be deterministic or stochastic. In the latter case the probability density functions pdf_X and pdf_D of X and D must be provided. Denote by $y_\Delta(x) = f_\Delta(x)$ the perturbed computation.

It is intuitive that when the size of the perturbation domain shrinks to zero the perturbed computation converges to

$$\lim_{\mathrm{Vol(D)}\to 0} y_\Delta(x) = y(x), \forall x \in X \tag{1.1}$$

where Vol denotes some statistical volume measure defined over D, e.g., Vol is function of the pdf_D's variance.

To characterize the accuracy of the approximated computation w.r.t the referenced one we have to compute the discrepancy between $y = f(x)$ and $y_\Delta = f_\Delta(x)$ by means of a suitable, but general, loss function

$$u(x, \Delta) = u(f(x), f_\Delta(x)). \tag{1.2}$$

The methodology must deal with the largest class of functions $f(x)$ to be applicable to a wide class of applications; this can be accomplished by requiring that $u(x, \Delta)$ is measurable according to Lebesgue with respect to (w.r.t.) X and D and that inputs and perturbations are stationary processes (their stochastic characterizations do not change over time).

Basically, all functions involved in signal/image processing are measurable according to Lebesgue [11]. In fact, all continuous functions and almost all noncontinuous nondifferentiable relevant functions related to sensors and data processing are measurable according to Lebesgue. Surely, the mean square error (MSE), the noise-to-signal ratio (NSR), the max function, all L_p norms and their compositions are measurable according to Lebesgue and constitute common examples for $u(x, \Delta)$. Fourier, Laplace, Wavelets, Cosine series and transforms, filters, neural networks, physical-based and identified models, just to name a few, their composition, and much more complex strongly nonlinear functions are measurable according to Lebesgue.

The proposed analysis for evaluating the performance loss $u(x, \Delta)$, or *verification problem*, can be divided into two different cases reflecting the applications needs.

The first verification problem addresses the case in which the reference computation is affected by a fixed perturbation $\bar{\Delta}$. The perturbation can be of any nature: structural modifications of the reference computation (we modify part of the algorithm which can be represented as a completely different function $y_\Delta = f_\Delta(x) = g(x)$), a given finite precision representation for some parameters of the algorithm, or removal of some computational units. Note that the first case is extremely general. Examples are the implementation of a nonlinear function with an approximated solution (e.g., Taylor or similar expansions), lossy transformations for data compression (e.g., wavelets), removal, or switching off some parts of the computation. The effect of a fixed perturbation is to bias the computation, which diverges from the nominal one for the systematic presence of a distortion; *de facto* the perturbed computation can be represented with a new function. No stochastic description is needed for the perturbation, which is given, fixed, and hence deterministic. The goal of the fixed perturbation verification problem is, therefore, to characterize the accuracy loss introduced by such perturbed computation.

The second verification problem addresses the case in which the perturbation Δ is not fixed but can assume all values within the dominion D. This problem is much more complex than the fixed perturbation one since, in general, D is continuous; in this case the probability density function pdf_D describes the probability that a value of the dominion D arises and affects the computation. In other words, this second problem addresses the case where the effect induced by a physical error is not known, apart from the fact it belongs to D.

Each perturbation has a different behavior being parameter specific. For instance, in the case of analog solutions, the production process of physical parameters (e.g., a resistance value) provides values subject to gaussian distributions [12], [13]; if we wish to study the ensemble behavior of a circuit we have to consider parameters varying in D. Variations due to aging and thermal effects represent another example where physical errors can be modeled and abstracted by perturbations belonging to D. In digital realizations perturbations can be associated with quantization effects, which can be nicely modeled as uniform perturbations defined over an interval [8]. With a suitable choice for D we analyze, in a unique solution, the possible effect of any physical error without the need for studying the effect of each on performance accuracy.

In several cases, we do not know anything about the distribution of the perturbation over D apart from the fact it is bounded. The designer can therefore select for such cases a uniform distribution. In fact, the uniform distribution considers all perturbations in the dominion to be equally probable and it is shown that its adoption is quite conservative, in the sense that it is more severe than many other distributions [14], [15].

A. *Verification Problem: The Perturbation Δ is Arbitrary, but Fixed*

The evaluation of the accuracy loss associated with the verification problem requires knowledge of $u(x, \Delta)$ which, for a generic Lebesgue measurable function $y = f(x)$ and a generic $y_\Delta = f_\Delta(x)$ is not available in a closed form. We can characterize the performance loss by following a different approach, which requires testing whether

$$u\left(x, \bar{\Delta}\right) \leq \gamma \text{ hold or not, } \forall x \in X \tag{2.1}$$

in correspondence with all positive values γ s. In other words, we are quantifying the loss in performance accuracy of the whole application. The analysis is completely different from [16] since (2.1) and the following derivations require, within a perturbed environment, investigations over the whole input space. Instead, the focus of [16] is on a fixed input set and, within such a framework, the impact of generic perturbations affecting the computational flow is studied.

We denote by $\bar{\gamma}$ the minimum value of γ for which (2.1) is fully satisfied, i.e., $u\left(x, \bar{\Delta}\right) \leq \bar{\gamma}$ holds $\forall x \in X$. $\bar{\gamma}$ identifies the maximum performance loss induced by $\bar{\Delta}$ on the computation and, hence, it provides an index of performance accuracy. Despite the fact that identification of the exact $\bar{\gamma}$ might be extremely difficult for a generic function, its value could be too conservative for subsequent analyzes. This aspect has been pointed out by other authors in problems related to the identification of the robustness margin index for robust control [17]–[19]. In fact, the maximum error is excited by particular perturbations which arise, in general, with a very low, almost null, probability. The risk of a deterministic worst case analysis is to be too conservative ($\bar{\gamma}$ is overdimensioned w.r.t. the application needs) with a subsequent inefficient use of resources. We will come back to this concept in later sections.

A dual probabilistic problem can therefore be formulated for (2.1), which requires that the perturbed computation satisfy, at least with probability η, a desired performance level γ. Denote by X_s the subset of X for which $u\left(x,\bar{\Delta}\right) \leq \gamma$ holds for a given γ. The probabilistic verification problem aims at computing the weighted volume of X satisfying (2.1) according to the probability density function

$$\Pr\left(x \in X_s\right) = \int_{X_s} \mathrm{pdf_X dx} = \Pr\left(\mathrm{u}\left(\mathrm{x},\bar{\Delta}\right) \leq \gamma\right). \qquad (2.2)$$

From (2.2) the definition of probably approximately correct computation (PACC) follows.

Definition: We say that a PACC is attained at level $\gamma > 0$ with probability η, when

$$\Pr\left(u\left(x,\Delta\right) \leq \gamma\right) \geq \eta, \forall x \in X. \qquad (2.3)$$

In other words, the probability that the performance loss is smaller than γ in X is greater than η. The probabilistic problem is weaker than the deterministic one and tolerates the existence of a set of inputs $\tilde{x}$ for which the bound associated with the loss in performance might not be satisfied ($u(\tilde{x},\Delta) > \gamma$); the probability of encountering such critical points is smaller than $1-\eta$. The rationale behind this is that if we introduce a performance loss γ we can guarantee that the perturbed computation does not introduce an error greater than γ for at least $\eta\%$ of inputs. Of course, we would rely on the computation only if 100% of inputs induce, in probability, an error below γ; this issue is formalized by the following definition.

Definition: We say that a computation is probably approximately correct at level γ and we denote by γ-PACC, when $\gamma = \bar{\gamma}$ is the smallest value for which

$$u\left(x,\Delta\right) \leq \bar{\gamma}, \text{ hold } \forall x \in X \text{ with probability one.} \qquad (2.4)$$

γ–PACC is a direct consequence of PACC when the probability η is one (all points in X satisfy the inequality $u\left(x,\Delta\right) \leq \bar{\gamma}$ with probability one).

The γ–PACC framework allows us to measure the performance loss associated with the perturbed computation. In fact, it states that the perturbed computation induces a performance loss smaller than $\bar{\gamma}$ with probability one; the set of points,if not empty, inducing a performance loss greater than $\bar{\gamma}$ has a measure according to Lebesgue null.

The reader could cast some doubts about the validity of results and worries about the use of a probabilistic approach to estimate the accuracy in performance (think of the case in which randomization is not extracting the worst perturbation). We first note that points inducing a loss in performance greater than $\bar{\gamma}$ could arise during the operational phase of the device with an almost null probability (are extremely rare). In addition, if the function we are considering is continuous with respect to X and D, so it is the $u(\Delta)$-transformed space. Under this assumption we have that points not satisfying the requirement, if any, lie close to the ones which satisfy it [38] and hence the estimate value is not far from the true value $\bar{\gamma}$. These comments have also been made in the robust control community [17], [18] in developing robust controls based on a probabilistic framework.

Therefore, the use of a probabilistic approach for verifying the performance loss associated with a perturbed computation does not constitute a limitation. Despite the evident advantages, which will arise in the next sections, we have to remember that the application field we are dealing with is the design of embedded systems for signal/image processing. In these applications a probabilistic setup is already implicitly hidden in the application. In fact, in most of embedded applications, inputs are signals and images [20] which are error affected, the parameters of the model have been identified [20]–[22] (and are therefore affected by a probabilistic uncertainty), the reference computation is an approximation of the "true" unknown computation [21], etc.

As an unorthodox and limit example to intuitively understand what is behind the theory and why a probabilistic approach is reasonable, we consider a mobile phone designed according to γ–PACC to tolerate a maximum performance loss $\bar{\gamma}$. x could be some voice-based signal or features processed by the device. In the reasonable case that pdf_X is continuous, an instance of x inducing a performance loss greater than $\bar{\gamma}$ will arise with low probability (possibly null) during the operational life of the device. Since its effect is to amplify the performance loss above $\bar{\gamma}$ it can be seen as an additional noise affecting the device. The user will not realize this error by claiming that the disturbance is due to external environmental conditions (e.g., electromagnetic distortions, low-power signal, etc) and by not blaming the device. Conversely, a more accurate $\bar{\gamma}$ will allow for a better dimensioning of the device by reducing silicon area and power consumption, hence yielding lower cost and longer battery life.

The following simple but relevant example evidences once more how the PACC philosophy is already intrinsic in almost all finite precision representation devices and how the probabilistic setup provides significant advantages to the deterministic one. The example shows that a deterministic knowledge of $\bar{\gamma}$ is too conservative while a probabilistic approach provides a much more useful estimate for subsequent dimensioning.

Example: Consider the reference computation associated with the scalar product evaluation $y = \sum_{i=1}^{n} x_i\theta_i$ (e.g., a linear filter [23]–[25]) of coefficients θ_i s. We assume, for simplicity, that no overflows/underflows occur, that truncation $\bar{\Delta} = [\delta\theta_1, \ldots, \delta\theta_n]$ affects the coefficients, and that n is large. To make the mathematics more amenable we also assume the inputs to be mutually independent and uniformly distributed in the [−1,1] interval. If we consider $u\left(\bar{\Delta}\right) = \delta y = \left(y - y\left(\bar{\Delta}\right)\right)$ as a loss function, the cumulated error is the random variable $\delta y = \sum_{i=1}^{n} x_i \delta\theta_i$.

In a deterministic setup, we have to consider the worst case analysis. The $x = x_1$ vector which maximally amplifies the error has components with absolute value 1 and opposite sign with respect to $\bar{\Delta}$

$$u\left(x,\bar{\Delta}\right) \leq \left(\sum_{i=1}^{n} |\delta\theta_i|\right) = \bar{\gamma}. \qquad (2.5)$$

The $\bar{\gamma}$ suggested by the deterministic setup identifies the maximum loss in performance we can expect at the device output.

The bound we have found, even if correct, is extremely conservative; we can identify a reasonable, less conservative, $\bar{\gamma}$ according to PACC.

From the central limit theorem [26] we have that δy tends to a gaussian distribution with zero mean and variance $E_x\left[\delta^2 y\right] = E_x\left[\left(x, \bar{\Delta}\right)^2\right] = \left(\left|\bar{\Delta}\right|^2\right)/3$. Please note that a common solution considered in the literature to dimension the variables involved in the linear computation at the bit level is based on a stochastic approach taking into account the variance value [5], [6] by means of a signal-to-noise ratio (SNR) figure of merit. Immediately, by choosing γ to be three times the standard deviation, we have that

$$\Pr\left(u\left(x, \Delta\right) \leq \sqrt{3}\left|\bar{\Delta}\right|\right) \geq 0.99$$

and the formulation coincides with the PACC one. For solving the γ-PACC problem we have to estimate the upper bound when η is close to 1; if we choose four standard deviations, η practically coincides with 1 and the associated $\bar{\gamma}$ is $\bar{\gamma} = 4/3\left|\bar{\Delta}\right|$.

To easily compare deterministic and probabilistic $\bar{\gamma}$ we can assume identical perturbations of magnitude $|\delta\theta_i|$. Therefore, the deterministic problem provides $\bar{\gamma}_d = n\left|\delta\theta_i\right|$ and the probabilistic one $\bar{\gamma}_p = 4/3\sqrt{n}\left|\delta\theta_i\right|$. The probabilistic setup has complexity $O\left(n^{1/2}\right)$ compared to the $O\left(n\right)$ one possessed by the deterministic counterpart and, hence, it is less conservative.

The provided information can be simply related to the synthesis phase by dimensioning the word length in a digital implementation. If we assume a fixed point representation for coefficients truncating the decimal part at q bits, we have that $|\delta\theta_i| \leq 2^{-q}$ from which $\bar{\gamma}_d = n2^{-q}$ and $\bar{\gamma}_p = (4/3)\ \sqrt{n}2^{-q}$; if we fix the tolerated loss in accuracy $\bar{\gamma}$ we can determine the required q. By considering the probabilistic bound instead of the deterministic one we immediately "save" $\log_2(\bar{\gamma}_d/\bar{\gamma}_p) \cong 0.5\log_2 n$ bits.

B. Verification Problem: The Perturbation Δ is not Fixed

The second performance verification problem addresses the more complex case where perturbations affecting the computation are not fixed, being variables defined over the D space. Obviously, the fixed perturbation verification problem is a subcase.

Different figures of merit can be envisaged to describe the computational loss associated with perturbations. We feel that a natural characterization covering a large spectrum of applications can be obtained by first averaging w.r.t. the perturbation space (so as to remove the dependency on Δ) and then consider the maximum error amplified by the inputs. More formally, the associated performance loss is

$$\tilde{\gamma} = \max_{x \in X} E_D\left[u\left(x, \Delta\right)\right]. \tag{2.6}$$

The chosen figure of merit must reflect the needs of the specific application. The designer could consider different max loss functions such as $\tilde{\gamma} = \max_{\Delta \in D} E_X\left[u\left(x, \Delta\right)\right]$ (the attention is focused on the maximum effect induced by perturbations on the averaged figure of merit w.r.t X), $\tilde{\gamma} = \max_{\Delta \in D} Var\left[u\left(x, \Delta\right)\right]$, or $\tilde{\gamma} = \max_{x \in X} Var_D\left[u\left(x, \Delta\right)\right]$ (the attention is on the maximum error of the energy of the loss function w.r.t the perturbation). Note that the analysis is again different from [16] since we are contemporarily exploring the X and D spaces.

Again, the main problem of (2.6) is related to the computational complexity needed to compute $\tilde{\gamma}$. In fact, a closed form solution can be obtained only in "toy" applications by considering simple loss and probability density functions. On the other hand, standard operational research tools cannot be implemented for a general Lebesgue-measurable function hence limiting their effectiveness. We can solve the verification problem by resorting to probability.

Definition: We say that a PACC is attained at level γ, accuracy ε, and confidence η for a $\mathrm{Max}\ u$ problem when the estimate of the maximum over M points $\hat{\gamma} = \max_{x_i, i=1,M} E_D\left[u\left(x, \Delta\right)\right]$ grants that

$$\Pr\left(\Pr\left(E_D\left[u\left(x, \Delta\right)\right] \geq \hat{\gamma}\right) \leq \varepsilon\right) \geq \eta. \tag{2.7}$$

The bound guarantees that the set of points x for which the expectation is greater than the estimated value $\left(E_D\left[u\left(x, \Delta\right)\right] \geq \hat{\gamma}\right)$ has a measure smaller than ε; such a statement is true at least with probability η. Of course a *Min* problem can be considered and immediately transformed into a *Max* problem.

Definition: We say that a computation is probably approximately correct at level γ, accuracy ε, and we denote by ε γ-PACC, when the estimate of the maximum over M points $\hat{\gamma} = \max_{x_i i=1,M} E_D\left[u\left(x_i, \Delta\right)\right]$ grants that

$$\Pr\left(\Pr\left(E_D\left[u\left(x, \Delta\right)\right] \geq \hat{\gamma}\right) \leq \varepsilon\right) = 1. \tag{2.8}$$

It is obvious that $\hat{\gamma} = \tilde{\gamma}$ satisfies (2.8) and, therefore, $\tilde{\gamma}$ is an ε γ-PACC index for the computational accuracy.

We introduce a toy example for which $\bar{\gamma}$ can be computed in a closed form; again we will discover that a probabilistic approach is more suitable than a deterministic one.

Example: Consider $y = ax$ to be the reference computation with a, x, and y real scalars. We choose a physical analog device in which a is subject to a gaussian distribution of $\bar{a}$ mean (e.g., $\bar{a}$ is the nominal value of a resistor). We assume also that x is drawn from a gaussian distribution with zero mean and χ variance. By choosing $u(x, \Delta) = (y - y(x, \Delta))$ we have that

$$E_D\left[u\right] = \bar{a}x$$

and the maximum value for $\tilde{\gamma}$ as suggested by the deterministic approach is plus infinity. Such a value is obviously not acceptable. Instead, by considering a probabilistic approach we have, at least with probability 0.99, that the required estimate is $\hat{\gamma} = 3\bar{a}\sqrt{\chi}$; this value can be used for subsequent device dimensioning as we did in the previous example.

The simple examples presented in Sections II-A and II-B have shown that a probabilistic approach is already present in several applications and that $\bar{\gamma}$ and $\tilde{\gamma}$ can be obtained in a closed form only in correspondence with toy examples supported by strong hypotheses to make the mathematics amenable. Estimates for $\bar{\gamma}$ and $\tilde{\gamma}$ can be derived by relaxing *ALL* hypotheses assumed in the literature and facing the problem with the recent theories based on randomized algorithms [19], [27]–[30].

Randomized algorithms are strictly related to the Monte Carlo method and turn, under weak hypotheses, an intractable problem into a tractable one, in our case with a polytime

complexity. The cost we have to pay is that results are valid in probability, with accuracy and confidence levels that can be made *arbitrarily* close to zero and 100%, respectively. Wide evidence for the effectiveness of such approaches can be found in the control theory community where great efforts have been devoted to the analysis and design of robust controllers [29]–[32].

III. Randomized Algorithms, γ–PACC and $\varepsilon\gamma$–PACC

In this section, we provide a methodology based on randomized algorithms for estimating $\bar{\gamma}$ and $\tilde{\gamma}$ as required in the PACC-based theories and show how it can be integrated within future CAD environments. The provided algorithms constitute the core of the performance degradation estimation methodology.

Denote by $p_\gamma = \Pr\{u(x,\Delta) \le \gamma\}$ the probability that the performance loss is satisfied at a given accuracy loss level γ.

The probabilistic approach needs the knowledge of p_γ which, in turn, requires exploration of the whole X space, a computationally hard problem. Since the curse of dimensionality would occur for any grid sampling on X [33], we resort to randomization by means of randomized algorithms.

A. Randomized Algorithms

Denote by u a generic loss function measurable according to Lebesgue with respect to the k-dimensional input space X and by pdf_x the associated probability density function. We assume that the process generating the data is stationary. Denote by Δ the fixed perturbation affecting the computation. Extract from X a set of N independent and identically distributed samples x_i according to pdf_x and generate the N triplets

$$\{x_i, u(x_i,\Delta), I(x_i,\Delta)\}, i = 1, N \tag{3.1}$$

where $I(x_i,\Delta)$ is the indicator function

$$I(x_i,\Delta) = \begin{cases} 1, & \text{if } u(x_i,\Delta) \le \gamma \\ 0, & \text{if } u(x_i,\Delta) > \gamma \end{cases} \tag{3.2}$$

and γ is a given, but arbitrary, positive value. The true unknown probability $p_\gamma = \Pr\{u(x,\Delta) \le \gamma\}$ can be estimated as

$$\hat{p}_N = \frac{1}{N}\sum_{i=1}^{N} I(x_i,\Delta). \tag{3.3}$$

It is intuitive that the adherence of $\hat{p}_N$ to p_γ depends on the required accuracy level ε so that $|\hat{p}_N - p_\gamma| \le \varepsilon$ and, indirectly, is function of the number of samples N we draw.

For its nature, $\hat{p}_N$ is a random variable and depends on the particular realization of the considered N samples (for each different set of cardinality N we obtain a different estimate for p_γ). This stochastic fluctuation can be tackled by resorting to probability and introducing a confidence degree $1-\delta$.

Finally, the Chernoff bound [34] provides the minimum number of samples N

$$N \ge \frac{\ln\frac{2}{\delta}}{2\varepsilon^2} \tag{3.4}$$

granting

$$\Pr\{|p_\gamma - \hat{p}_N| \le \varepsilon\} \ge 1-\delta, \forall\gamma \ge 0, \forall\delta,\varepsilon \in (0-1). \tag{3.5}$$

In other words, by considering the number of samples given by (3.4), (3.5) asserts that "$\hat{p}_N$ approximates p_γ with accuracy ε" and the statement is true at least with probability $1-\delta$.

Other bounds can be considered instead of the Chernoff one (e.g., the Bernoulli and Bienaymè ones, e.g., see [19]). Nevertheless, the Chernoff bound improves upon the others since it requires extracting the minimum number of samples. The Chernoff bound is particularly interesting since the number of points to be extracted is independent from the dimension of D (and hence it *does not* depend on the number of perturbations we are considering). In addition, the number of points required to explore the space is polynomial in the accuracy and confidence degrees; if $u(x,\Delta)$ can be evaluated in a polytime so can the PACC problems.

B. Procedure Based on Randomized Algorithms for Solving the γ-PACC Problem (Δ is Arbitrary but Fixed)

The evaluation of the performance loss $\bar{\gamma}$ characterizing the accuracy of the computation identified by γ-PACC can be estimated with accuracy ε and confidence level $1-\delta$ by means of randomized algorithms. In fact, from (3.3) and (3.5) we have that

$$\begin{aligned}\Pr\{|p_\gamma - \hat{p}_N| \le \varepsilon\} \ge & 1-\delta \\ \equiv & \Pr\{|\Pr(u(x,\Delta) \le \bar{\gamma}) \\ & - \frac{1}{N}\sum_i I(x_i,\Delta)\Big| \le \varepsilon\Big\} \\ \ge & 1-\delta.\end{aligned} \tag{3.6}$$

From (3.6), provided that ε and δ are small enough, we can confuse p_γ and $\hat{p}_N$ and (2.3) becomes

$$\Pr(u(x,\Delta) \le \gamma) \ge \eta \underset{\varepsilon\longrightarrow 0}{\cong} \hat{p}_N \ge \eta. \tag{3.7}$$

Therefore, the PACC problem immediately derives and requires testing whether the relationship $\hat{p}_n \ge \eta$ is satisfied or not. The γ-PACC problem follows directly from its definition. In fact, when η tends to one we can identify the $\hat{\gamma}$ associated with the performance loss of the perturbed computation

$$\Pr(u(x,\Delta) \le \hat{\gamma}) \cong 1 \Leftrightarrow \hat{p}_N \cong 1 \Rightarrow \hat{\gamma}. \tag{3.8}$$

The final algorithm for determining the estimate of $\bar{\gamma}$ is given in Fig. 1.

The obtained $\hat{\gamma}$ is an index of the computation accuracy for the embedded system once affected by the perturbation. If device D_1 (perturbed computation 1) introduces a performance loss $\hat{\gamma}_1$ and D_2 (perturbed computation 2) a loss $\hat{\gamma}_2$, D_1 is more accurate in the computation than D_2 if and only if $\hat{\gamma}_1 < \hat{\gamma}_2$.

Since most of computations in embedded systems are characterized by polynomial complexity algorithms, thanks to Chernoff, the computational accuracy degradation can be estimated with a polytime algorithm as well.

CHARACTERISE X;

/* IDENTIFY THE ACCURACY AND THE CONFIDENCE LEVELS FOR THE APPLICATION */

SELECT ε, δ;

/*TEST THE γ -PACC FOR THE COMPUTATION */

EXTRACT $N \geq \frac{\ln\frac{2}{\delta}}{2\varepsilon^2}$ POINTS FROM X ACCORDING TO pdf_X;

EVALUATE $\{u(x_i,\Delta), I(x_i,\Delta)\}, i=1,N$;

GENERATE THE FUNCTION $\hat{p}_N = \hat{p}_N(\gamma), \forall\gamma \geq 0$;

SELECT THE MINIMUM VALUE $\hat{\gamma}$ SO THAT $\hat{p}_N(\hat{\gamma}) = 1, \forall\gamma \geq \hat{\gamma}$;

$\hat{\gamma}$ IS THE ESTIMATE OF $\tilde{\gamma}$.

Fig. 1. The procedure for estimating the performance loss $\hat{\gamma}$.

C. Procedure Based on Randomized Algorithms for Solving the γ-PACC Problem (Δ is not Fixed)

An estimate for $\tilde{\gamma}$ as required in (2.6) can be obtained by sampling the input space according to pdf_X. If we consider M samples, an estimate of the maximum can be obtained as

$$\hat{\gamma} = \max\{E_D[u(x_1,\Delta)],\ldots,E_D[u(x_M,\Delta)]\}. \quad (3.9)$$

For a fixed input x_i, the exact computation of $E_D[u(x_i,\Delta)]$ requires the solution of the integral associated with the expectation and, in general, is very difficult. We resort then to randomization. Different from the fixed perturbation performance verification case, here, we have two degrees of freedom which make the analysis more complex. The problem is close to the one discussed in [19] and [35] for obtaining a probabilistic robust control design, provided that the input x becomes the controller. Draw then M samples from X and N from D according to the respective *pdf*s; for any $\varepsilon \in (0,1)$ and $\delta \in (0,1)$ and with the samples choice

$$M \geq \frac{\ln\delta}{\ln(1-\varepsilon)} \text{ and } N \geq \frac{\ln\frac{2M}{\delta}}{2\varepsilon^2} \quad (3.10)$$

we have that

$$\Pr\left\{\Pr\left\{E_D[x,\Delta] \geq \max_{i=1,\ldots,M}\hat{E}_D[x_i,\Delta]+\varepsilon\right\} \leq \varepsilon\right\} \geq 1-\frac{\delta}{2} \quad (3.11)$$

holds. Equation (3.11) states that "the set of points greater than the estimated value $\hat{\gamma}$ has a measure smaller than ε"; the statement is true at least with probability $1-\delta/2$. This implies that if the function $E_D[u(x,\Delta)]$ is sufficiently smooth then the estimated value $\hat{\gamma}$ and the actual one $\tilde{\gamma}$ are very close; we experienced that this holds in all the envisaged applications as also verified in [19]. When δ and ε are sufficiently small (3.11) becomes a nice approximation of the (2.8) and we obtain the $\varepsilon\gamma$–PACC formulation. The algorithm to compute $\hat{\gamma}$ is finally given in Fig. 2.

CHARACTERISE X AND D;

/* IDENTIFY THE ACCURACY AND THE CONFIDENCE FOR THE APPLICATION */

IDENTIFY ε, δ;

/*TEST THE $\varepsilon\gamma$ -PACC OF THE SOLUTION */

SELECT $M \geq \frac{\ln\delta}{\ln(1-\varepsilon)}$ AND $N \geq \frac{\ln\frac{2M}{\delta}}{2\varepsilon^2}$;

EXTRACT M POINTS x_i FROM X ACCORDING TO pdf_X;

EXTRACT N POINTS Δ_k FROM D ACCORDING TO pdf_D;

COMPUTE THE M VALUES $\hat{\gamma}_i = \frac{1}{N}\sum_{k=1}^{N} u(x_i,\Delta_k)$;

SELECT THE ESTIMATE FOR $\tilde{\gamma}$ AS $\hat{\gamma} = \max\{\hat{\gamma}_1,\ldots,\hat{\gamma}_M\}$;

Fig. 2. The procedure for selecting $\hat{\gamma}$ according to $\varepsilon\gamma$-PACC.

D. Integrating the PACC Methodology in Future CAD Environments

The two verification problems constitute the core of the methodology for estimating, at high level, the performance degradation of an embedded system characterized by a Lebesgue-measurable computation.

To apply the methodology, the designer has to provide:

1) the reference computation described by means of a high-level formalism (e.g., a data flow description carried out with a C-like language);
2) the stochastic description of the input space X (or a number of samples as required by the Chernoff bound extracted according to pdf_X);
3) the figure of merit measuring the performance loss, i.e., the discrepancy between the reference computation and the perturbed one;
4) the perturbed computation in the fixed perturbation case or the perturbation injecting points and the stochastic description of D in the nonfixed perturbation case.

At this level, the main difference between the fixed and the nonfixed perturbation cases is only in point 4. In more detail, if

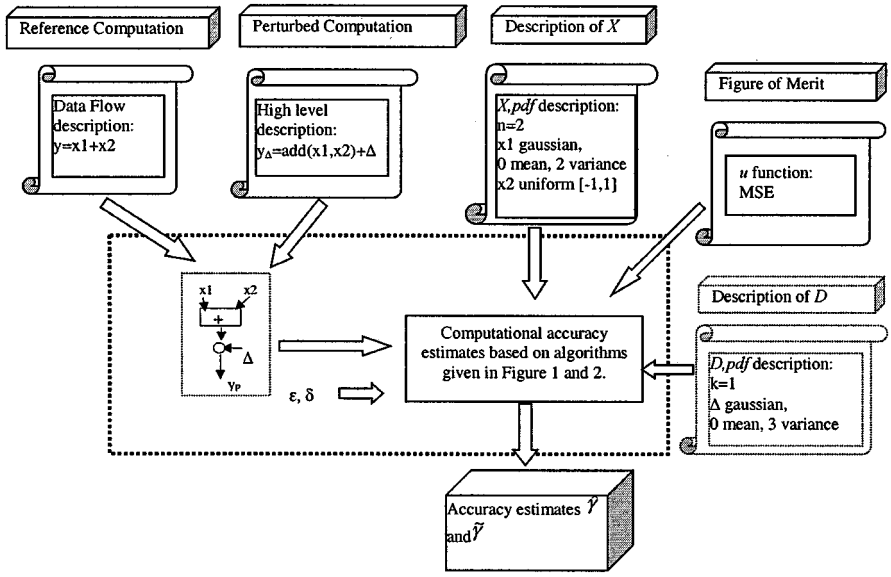

Fig. 3. The high-level structure of the methodology applied to a simple example.

the perturbation is fixed what it is needed is solely the function describing the perturbed computation (i.e., the embedded solution) and we are asked to validate it by testing the discrepancy between reference and perturbed computations. The verification aspect is carried out in SW by applying the algorithm given in Fig. 1.

The second verification problem requires knowledge about the placement of the perturbation injection points along the reference computation and a stochastic description for each component of the perturbation vector D. The perturbation injection points are identified by the designer on the reference computation on the basis of a first high-level design of the solution. The designer decomposes this embedded solution prototype in modules and introduces at the output of a generic module an independent perturbation where it is expected that the subsequent physical implementation will introduce an independent source of error. A perturbation must, in fact, abstract low-level sources of uncertainties by means of its behavioral description (please also refer to the experimental section for an example showing how to partition the application solution in modules and introduce appropriate perturbations). To this end, we also note that effects related to the propagation of errors along the computational chain do not constitute an independent source of error and therefore no perturbation variables are needed to represent such effects which are directly considered by the methodology. Also note that perturbations are not emulated in HW but simulated in software as random extractions from a *pdf*.

The high-level structure of the methodology is depicted in Fig. 3. By receiving application level descriptions about the reference computation and the perturbed computation, the methodology provides an effective measure of the computational accuracy for the application.

By referring to the simple example given in Fig. 3, the designer wishes to measure the performance loss associated with an embedded device whose task is to add inputs $x1$ and $x2$. The designer is assuming that the input variables are not affected by errors and that the unique source of error will be associated with the realization of the addition operator. The designer has in mind an analog implementation and models the error affecting the adder as subject to a gaussian *pdf* (e.g., he characterizes the tolerance of a resistor according to the production specifications). Of course, we could have considered the different description $y_\Delta = add(x1, x2+\delta x_2)+\delta_{\rm add}$, hence assuming that also variable $x2$ is affected by perturbations (e.g., finite precision representation) in addition to the adder one. More in general, the designer, having in mind a first prototypal architecture, introduces the independent perturbations at the output of the modules which will be locally affected, during implementation, by physical sources of uncertainty.

The mechanism described in Fig. 3 can now be integrated within a CAD environment as depicted in Fig. 4 where a high-level iterative synthesis phase is presented. We denote by "abstract architecture" a perturbed computation obtained by locating the perturbation injection points on the reference architecture; in a sense, an abstract architecture abstracts a set of physical ones by means of perturbations. For instance, the abstract architecture $y_\Delta = {\rm add}(x1, x2+\delta x_2)+\delta_{\rm add}$ abstracts all implementations introducing uncertainties on x_2 and the adder (e.g., finite precision representation/quantization) independently from any technological details (for a deeper analysis refer to [16]). If the abstract architecture refers to a fixed perturbation case, then the perturbations present in the abstract architecture are given, e.g., $y = trunc({\rm add(x1, trunc\,(x2)))}$ where ${\rm trunc}$ is a truncate operator acting at some fixed bit level.

Let us analyze in more detail the synthesis flow of Fig. 4 by focussing the attention on the nonfixed perturbation case; extensions to the fixed perturbation case are immediate. In the figure, continuous lines define the more natural path, dashed lines constitute alternatives which can be taken into account by the synthesis strategy.

At first, the designer defines the nature of the application inputs, the reference computation, and the performance loss figure of merit. Afterwards, the designer, having in mind a first high-level design for the embedded system, suggests an initial abstract architecture by defining the perturbation injection points on the reference computation and an initial value for D.

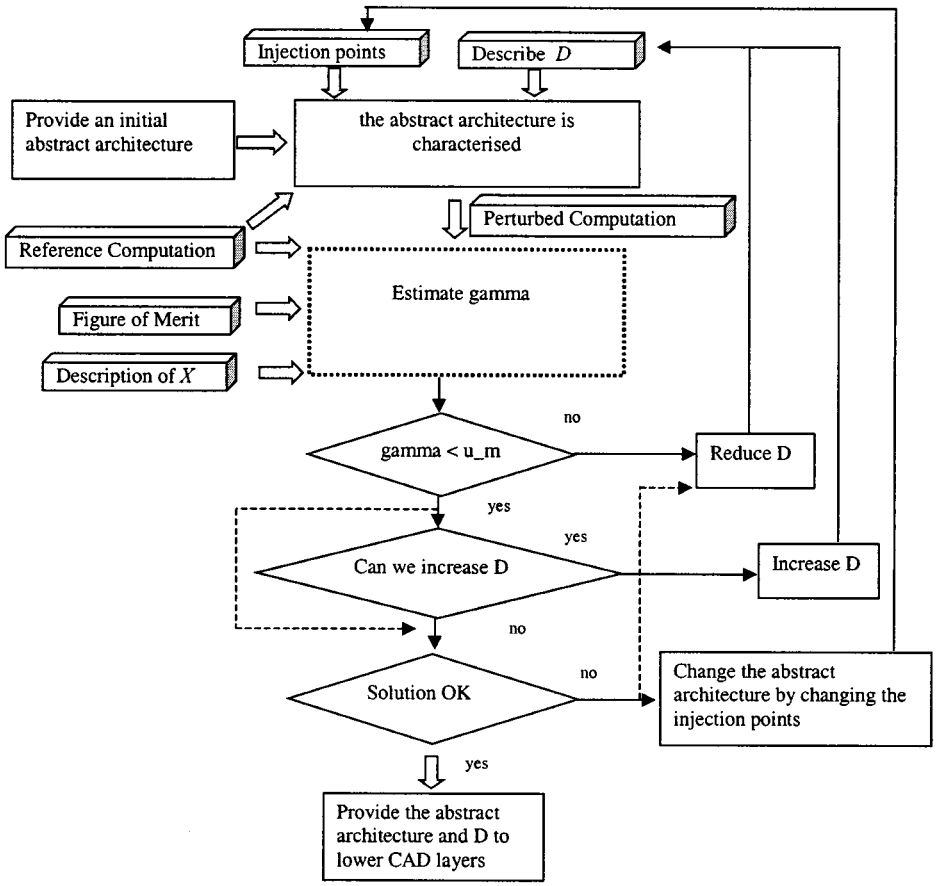

Fig. 4. The high-level structure of the methodology and the interaction with a CAD environment.

In turn, the perturbation injection points and the complete stochastic characterization of D define the perturbed computation. At this point the performance loss estimation core of Fig. 3 can be activated and provides an effective performance loss "gamma." If gamma is below a tolerated performance loss for the application "u_m," then the perturbed computation is feasible for the application needs at least w.r.t. the accuracy requirement. If not, it simply means that the perturbed computation is not robust enough and that the perturbation strength, characterized by $\mathrm{vol}(D)$, must be reduced (e.g., by increasing the variance of a gaussian distribution or the extreme of the interval for a symmetrical uniform distribution; see also [16]). This can be accomplished by reducing the volume of D. It is interesting to note that when the $\mathrm{vol}(D)$ satisfying the application performance requirement is extremely small the abstract architecture can tolerate small perturbations. The associated module requires high resolution and a floating point representation might be needed. Conversely, if the perturbed application is feasible, it can be interesting to identify the larger D still granting for an acceptable performance loss (if not we follow the dashed line); this can be accomplished by enlarging the volume of D. When we have identified the maximum D still granting for an acceptable loss, we have to verify if other application constraints are satisfied (e.g., we could move to lower levels of the synthesis phase, simulate a final implementation, and test whether power consumption and silicon area constraints are satisfied). If not, we have to think about reducing D (dashed line) or providing a new candidate architecture for the embedded system. The latter selection will end up with a new abstract architecture (at this level we have simply to change the perturbation injection points on the reference computation to abstract a new physical architecture). Finally, when the solution is acceptable, we will provide the perturbed computation and D to the lower levels of the CAD environment. In fact, from the abstract architecture we can directly describe the structure of the final architecture, e.g., by representing in VHDL the modules composing the computational flow. The indications coming from D will be used to synthesize the module; *whatever the final implementation will be, the errors introduced by the implementation of the module must belong to* D. In our simple case $y_\Delta = \mathrm{add}(x1, x2{+}\delta x_2){+}\delta_{\mathrm{add}}$, let us assume that D is characterized by a symmetrical uniform distribution and that the largest perturbation domain for δx_2 and δ_{add} have extreme values α_{x1} and α_{add}, respectively. From [8], and by considering truncation, we have that $x1$ and y can be truncated by removing the bits whose weight is below $\log_2 \alpha_{x1}$ and $\log_2 \alpha_{\mathrm{add}}$, respectively. We have, at this point, all the elements necessary to characterize the circuit in VHDL. A more complete formalization of the synthesis algorithm is beyond the goals of this paper and is currently being studied by the author; the next section will show how the methodology can be applied to support the designer in real embedded systems.

IV. Experimental Results

In the experimental section, we present two examples to show how the PACC methodology can be used to estimate the per-

formance loss of an application. The first application refers to a core component of an optical character recognition (OCR) system for a fixed style decimal digit classification, the second to a wavelets-based algorithm developed for image compression/processing.

A. A Neural Network-Based Application

A nontrivial algorithm based on a neural network classifier has been developed to cope with the nonlinear noisy environment (noise can be generated by fax transmission and/or scanner devices and significantly corrupts the digits).

The algorithm solving the application receives a 10×10 pixel matrix containing the digit as provided by an image segmentation phase and classifies the digit accordingly. The neural classifier has been optimally dimensioned w.r.t. performance and complexity [21], [22].

The final neural structure has 100 inputs, 30 hidden units, and a single output neuron which ends in a complex nonlinear computation. We wish to compute the performance degradation of the algorithm when the 3030 weights of the network are simultaneously perturbed. The loss in classification (accuracy) performance is due to propagation of the perturbation effect up to the classifier output.

We have to characterize first the inputs of the robustness layer as required in Fig. 3. In the specific application we have the following.

1) *Reference computation*: the reference computation is the neural computation implemented by the nonlinear neural classifier.
2) *Perturbed computation*: the neural computation perturbed by the presence of disturbances affecting the network weights.
3) *Description of* X: $n = 100$ (we have a 10×10 one-bit pixel matrix). The distribution is uniform, in the sense that an equal number of inputs is considered for each digit class (each digit has the same probability to be generated). Each image appears as provided by a scanner device.
4) *Figure of merit*: the considered figure of merit is the mean classification error defined over N inputs as $u(x, \Delta) = N^{-1} \sum_{i=1}^{N} |y(x) - y(x_i, \Delta)|$; the ($|y(x) - y(x_i, \Delta)|$ function assumes value 0 if pattern x_i is correctly classified, 1 otherwise).

We examine now the two perturbation cases identified in Section III.

The Fixed Perturbation Case: We consider the weights of the neural network perturbed by a fixed perturbations $\bar{\Delta}$ which affects simultaneously the weights of the network according to the multiplicative model $w_p = w(1 + \sigma)$; for instance, $\sigma = 0.4$ means that each weight is perturbed with a value up to 40% of its magnitude. The situation models the case where the chosen and given implementation introduces an error only on the network weights. We applied the fixed perturbations by adding a randomly extracted value to the perturbation injection points and we consequently tested the computational accuracy by considering the algorithm given in Fig. 1. The performance loss function $\hat{\gamma} = \hat{\gamma}(\sigma)$, obtained by considering stronger perturbations, is given in Fig. 5.

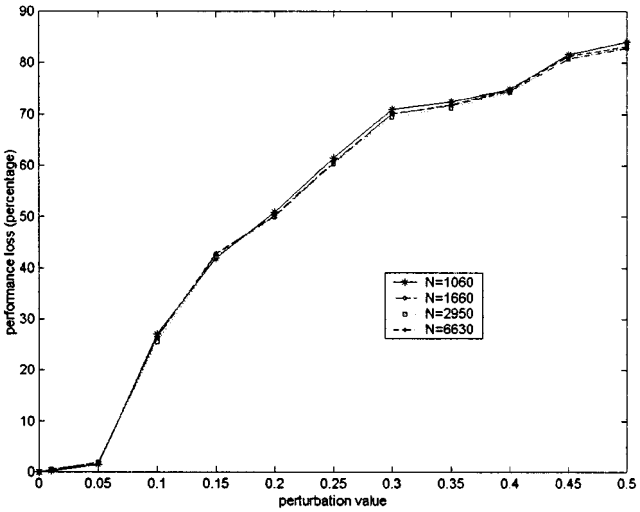

Fig. 5. The performance loss $\hat{\gamma} = \hat{\gamma}(\sigma)$ function for the fixed perturbation case.

While the performance loss $\hat{\gamma} = \hat{\gamma}(\bar{\sigma})$ for a given perturbation provides a measure of accuracy for the given computation only in correspondence with $\bar{\sigma}$, the $\hat{\gamma} = \hat{\gamma}(\sigma)$ function the performance loss function for a set of perturbed computations, hence somehow representing an accuracy signature for the application. Such information can be used during the synthesis phase, e.g., by identifying on the curve an acceptable performance loss from which we select the final realization.

Fig. 5 has been generated by considering a fixed value $\delta = 0.01$ for the confidence parameter (i.e., 99% of confidence) and different accuracy levels ($\varepsilon = 0.05$, 0.04, 0.03, 0.02 associated with $N = 1060$,1660, 2950, 6630, respectively).

As a first note, we observe that the application is rather insensitive to accuracy and confidence parameters since the curves do not vary significantly with N. Therefore, we should consider the lowest value of N for testing the accuracy of the computation since it will grant reliable results. This comment is of fundamental importance to reduce the computational burden associated with the estimate of the accuracy loss for a class of applications.

Fig. 5 shows that the application is very sensitive to multiplicative perturbations of some relevance. Once the magnitude of the fixed perturbation increases above 0.05, we experience a significant loss in classification performance for the given implementations. This is not surprising since the weights of the neural network constitute the "knowledge space" of the model.

The obtained information about accuracy immediately tells us that we have to represent weights with high accuracy to keep the loss in performance small. For instance, a finite precision representation for the weights must not introduce a multiplicative error bigger than 0.05 if we want to keep the additional performance loss of the classifier smaller than 3%.

The Nonfixed Perturbation Case: The nonfixed perturbation case models the realistic situation in which the network weights need to be represented in a finite precision representation and, hence, perturbations span a dominion D (the one abstracting and hiding the particular realization associated with finite precision representations). Differently from the fixed perturbation case,

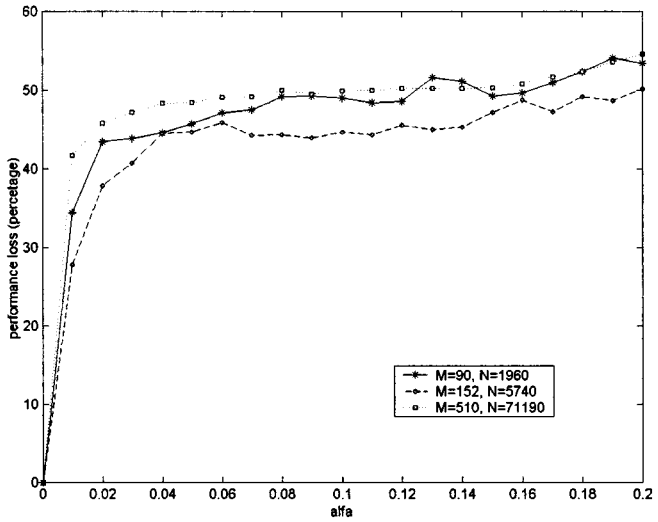

Fig. 6. The performance loss $\hat{\gamma} = \hat{\gamma}(\sigma)$ function for the additive nonfixed perturbation case.

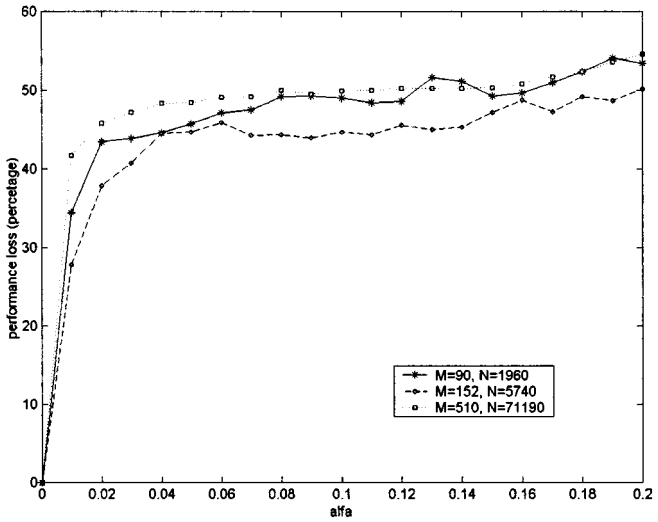

Fig. 7. The performance loss $\hat{\gamma} = \hat{\gamma}(alfa)$ function for the multiplicative nonfixed perturbation case.

we have to characterize the perturbations affecting the computation, here by assuming that the perturbation space is driven by a zero mean gaussian distribution for additive perturbations and a zero mean uniform distribution for multiplicative perturbations.

In fact, a gaussian additive perturbation nicely models an analog representation where the production process of a component is ruled by a gaussian distribution with zero mean and a given variance σ^2; such a perturbation is additive to the nominal value of the parameter [13], here representing the weight.

Conversely, a digital representation for the weights introduces errors which can be modeled as being extracted from a uniform distribution [8]. In this case, it is reasonable to model the perturbations as proportional to the weight magnitude (multiplicative model). While a gaussian process is characterized by mean and variance, a uniform distribution is characterized by the extremes of the perturbation interval. To formalize quantization techniques [8] we consider symmetrical intervals of extreme $alfa$. For instance, due to the multiplicative error model, an $alfa = 0.1$ means a maximum of 10% percentage error affecting the weights magnitude. It is interesting to observe that $alfa$ controls the volume of the perturbation space. The final loss function is the one suggested in (2.6) .

Figs. 6 and 7 present the $\hat{\gamma} = \hat{\gamma}(\sigma)$ and $\hat{\gamma} = \hat{\gamma}(alfa)$ gamma functions as identified by the PACC theories and, in particular, by the algorithm given in Fig. 2. For each plot we considered three runs with a fixed confidence level $\delta = 0.01$ (99% of confidence) and different accuracy levels $\varepsilon = 0.05$, 0.03, 0.009 ($M = 90$, $M = 152$, $M = 510$, respectively). From the plots we can see that the network is sensitive to small perturbations associated with both perturbation models and, therefore, the perturbed computation is rather critical in its finite precision representation. Nevertheless, the designer might accept a loss in performance, let us say 10% if this is paid back by a low cost (less accurate) analog component for the weights or a reduced number of bits for registers/memory (which implies area and power consumption).

In addition, the designer, by exploring the structure of the $\hat{\gamma} = \hat{\gamma}(\sigma)$ and $\hat{\gamma} = \hat{\gamma}(alfa)$ gamma functions realizes that the performance degradation is very rapid with $\mathrm{vol}(D)$. Therefore, by referring to Fig. 4, testing a new solution with an increased volume D (which will lead to an additional bit saving for a digital implementation) would reasonably not provide a tolerable loss in performance. Conversely, if the curves increase slowly it means that the application will be able to tolerate stronger perturbations (larger $\mathrm{vol}(D)$) and the designer will investigate more compact/less power-demanding solutions.

Extensions to consider mixed solutions are immediate and require a different characterization of the perturbation space as done in the last experiment. For the neural network, if a weight is represented with an analog solution we will consider an additive perturbation model with a gaussian pdf_D, in the digital counterpart the model must be multiplicative with perturbations extracted from a uniform distribution. The methodology is technology independent and, therefore, integration of analog and digital parts of the computation can be done directly at a system level by acting on the perturbation nature.

B. Wavelets-Based Application

The second application refers to the development of an embedded system for a quality analysis of steel cutting with laser technology. The embedded system could be mounted directly on the optical head for an online processing and transmission of retrieved sensorial information to external storage systems. The still-under-investigation application has been carried out with Trumpf GmbH, Germany. We have discovered that the evolution of the sparks jet over cutting time can solve the quality analysis problem. A set of features can be extracted from each retrieved image frame of sparks by means of wavelets transforms, while a classifier provides quality analysis indications of the cut artefact. For its large interest in embedded systems, and in particular in compression applications, here we focus the attention on the wavelets core. As with the previous experiment, the inputs required by the methodology are as follows.

1) *Reference computation*: the reference computation is a two-dimensional discrete wavelets transform (DWT) [36] applied to the image pixels.

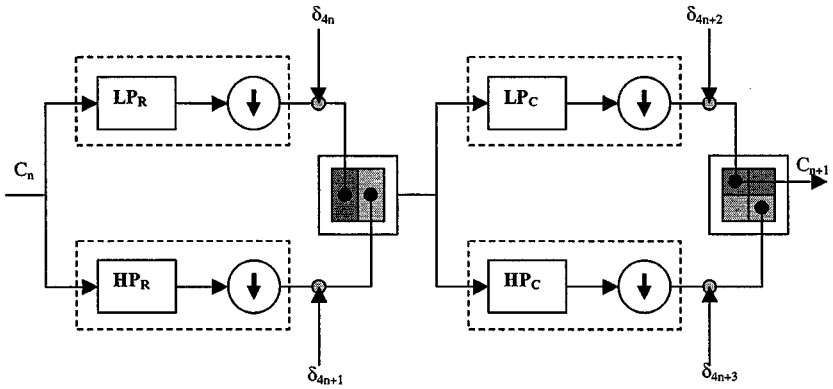

Fig. 8. The reference computation, the identified subsystems, and the perturbations.

2) *Perturbed computation*: the DWT computation in which the output of each filter + decimation module is affected by perturbations.
3) *Description of* X: each input is a frame containing the sparks jet associated with the cutting process (256×256 grayscale pixel resolution). Its *pdf* is the real one coming from the cutting process, e.g., we have to consider a set of images coming from the process (the cardinality of such set must satisfy the Chernoff's bound).
4) *Figure of merit*: we considered the mean square error between the original image and the perturbed one once processed by the inverse wavelets error-free transformation.

As with [37] the 2-D DWT coefficients can be obtained by considering a 2-D filter bank with decimation and biorthogonal wavelets, associated with 7 and 9 taps filters for the high-pass and the low-pass subbands. The pixels composing the image are first processed by rows with the 1-D filter and the outcome is further processed columnwise through an identical filter.

A four-level multiresolution analysis has been envisaged for the application (the filtering procedure must be iterated four times to generate approximation and detail features). The high-level structure of the computation is shown in Fig. 8 where LP and HP stand for low- and high-pass filters, respectively. The input image C_n is filtered by rows (left segment of the computation) and the output is subsampled with a decimation factor 2:1 (the downward arrows). Afterwards, the transformed image is filtered columnwise. This process iterates four times with the new input image being the output C_{n+1} of the previous iteration.

We suppose that the designer wishes to study, at the very high level, an architecture in which each module (dashed rectangle) is an independent processing unit. Nothing is said about technology or the way modules will be internally configured or implemented. The high-level abstract architecture to be tested is therefore composed of 16 independent processing modules. It should be noted that the designer could have considered a different folded abstract architecture, composed of a unique functional unit implementing the 16 processing ones; of course the perturbation model and hence the abstract architecture would have been different.

At the output of each subsystem the designer injects an independent perturbation variable modeling the fact that each module will be affected by physical errors. A uniform distribution is considered for each perturbation for its conservative nature as explained in Section II. Since there are $k = 16$ independent subsystems we must consider 16 independent perturbations affecting simultaneously the computation. The perturbation space is therefore $D = \{\delta_{4n+j}\}, \delta_{4n+j} \in [-\alpha_{4n+j}, \alpha_{4n+j}]$ where $n = 0, 1, 2, 3$ indicates the considered transformation level, $j = 1, 2, 3, 4$ indicates the addressed 1-D filter, and δ_{4n+j} is a generic independent perturbation extracted from a zero-mean, symmetrical, and uniform distribution of extreme α_{4n+j}. Chernoff grants that results are independent from the number of addressed perturbation injection points. The software simulation can then be run to provide the accuracy degradation indexes for the perturbation cases.

The Fixed Perturbation Case: We consider a case in which the designer wants to test the impact of truncation-like operators applied to the output of each module. In a way, the designer wishes to test a possible digital implementation where the unique source of perturbation affects the output of each computational module (e.g., truncation of the final accumulator value which impacts on the dimension of the word to be passed to the next computational unit). Again, note that the analysis is done at the application-level (e.g., in C language); no VHDL or VERILOG code needs to be written. The presence of a fixed perturbation modifies the reference computation, which is distorted. Even if the perturbation is fixed, we have that different inputs will generate different errors at that perturbation injection points. To keep an immediate close relationship to the digital world, we considered a set of interesting fixed perturbations associated with truncation. In particular, we consider the case in which truncation acts and keeps bits whose weight is at least 2^n (i.e., $n = 0$ means that we truncate the fractional part of the ideal value, $n = 2$ means that the weight of the first bit is 4). If the integer part of the signal (the error free output) can be represented with l bits in a two's complement notation, we have that by considering truncation at n we can represent the output value with $l - n$ bits.

The loss in performance associated with such perturbation is measured by the MSE loss function as given in Fig. 9 where two experiments with $\varepsilon = 0.05$, 0.03, and $\delta = 0.01$ are presented.

We note that there is no relevant difference between the two curves. This means that what we have obtained maximum accuracy of 100% since results are not strongly dependent on ε. From experiments we discovered that the embedded application can support a performance degradation up to $MSE = 400$. As a

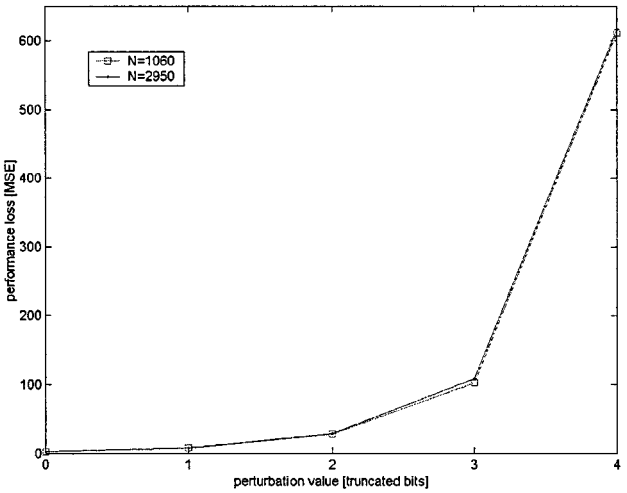

Fig. 9. The performance degradation for the fixed perturbation case.

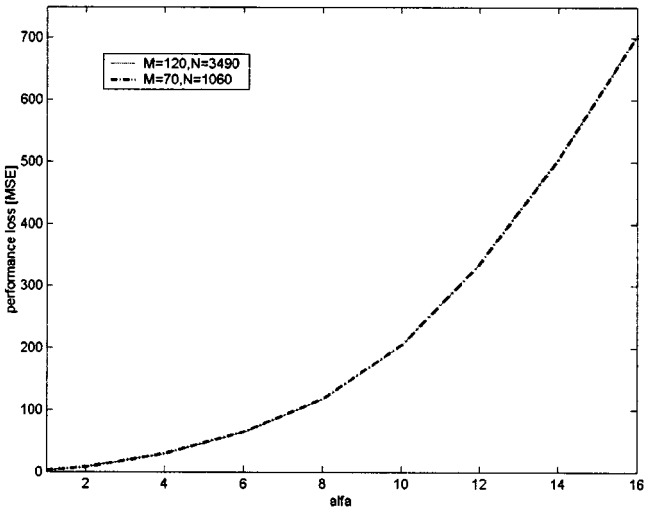

Fig. 10. The performance degradation for the nonfixed perturbation case.

consequence, the integer part of each module output can be represented with l-3 bits (the fractional part is not relevant at all) with a positive impact on memory size and power consumption. As we already mentioned, in this very simple case the theory is immediately supported by existing analysis CAD tools. Nevertheless, with respect to them here we provide a main advantage: we can state that the performance–accuracy loss index we have identified is the correct one with 100% accuracy and a confidence value of 99% ($\delta = 0.01$). Such a result is now reliable and can be used safely in subsequent design steps. In addition, we are sure that results are not biased to the particular set of inputs since we have extracted them from pdf_X as suggested by the theory.

The Nonfixed Perturbation Case: As we did in the first experiment, we consider now the nonfixed perturbation case. The problem addresses the situation in which the output of the module composing the abstract architecture can be affected during its operational life by any possible perturbation, provided it is defined in D. The analysis is hence no more limited to a specific perturbation (e.g., the one induced by truncation as we did in the previous experiment). Now, we are dealing with all quantization operators, all architectures implementing the module, and any source of error affecting internally the module (provided that its effect at the module output belongs to D). This comment has a fundamental impact on the synthesis phase. In fact, if the Ith module has associated the tolerated perturbation space D_I, then we can implement the module as we please with any architecture, any technology, any finite precision implementation, provided that its effective error at the module output belongs to D_I. D_I represents the tolerable perturbation space for which the performance degradation is acceptable.

For simplicity, we assumed that each of the 16 modules is subject to perturbations uniformly extracted from the same $[-alfa, alfa]$ interval (of course we could have considered different extremes). With such a choice $alfa$ is the unique parameter ruling the volume of the perturbation space. We considered two experiments with $\varepsilon = 0.067$, $\varepsilon = 0.038$, and $\delta = 0.01$ from which we selected $M = 70$ and 120, $N = 1068$ and $N = 3490$, respectively. The performance degradation function is given in Fig. 10. Since the application provides a tolerable loss in performance for perturbations with MSE below value 400, we can surely tolerate any perturbation belonging to the [−12, 12] interval; we have characterized D_I for the subsequent synthesis phase. Again, what we are asserting is true with probability 0.99 and 100% in accuracy since the two plots are almost coincident. The designer can now open the black box associated with the module and think of the subsequent internal implementation at the application level. Any solution will be acceptable provided that the induced performance loss belongs to the [−12, 12] interval. Of course, the designer can apply again the methodology to the architecture composing the module and test its performance degradation with the suggested methodology. Injection points, reference perturbations, will now be tailored to the filter + decimation computation according the architectural solution under test.

V. Conclusion

The paper introduces a methodology to estimate the performance degradation of an embedded system once subject to perturbations. Perturbations are abstractions of physical sources of uncertainties and, as such, hide all technological low-level details. For instance, perturbations can be associated with finite precision representations, parameter fluctuations due to the production process, or battery power variations. To cover a large spectrum of signal and image processing applications we removed all the hypotheses presented in the sensitivity literature by only requiring that the figure of merit measuring the performance loss is measurable according to Lebesgue. Two definitions for a probably correct computation have been introduced which quantify the impact of perturbations on performance with a polynomial time complexity. The derived performance–accuracy indexes constitute relevant information which can be used in subsequent design steps.

Acknowledgment

The author wishes to thank Prof. M. Sami for her help and the fruitful discussion during the preparation of the manu-

script, Trumpf GmbH for having provided the frames, and the reviewers whose insights and hints significantly helped to improve the manuscript.

REFERENCES

[1] Synthesis Tools, Ver. 2000.05–1 ed., 2000.

[2] G. D. Micheli and R. K. Gupta, "Hardware/Software Co-Design," *Proc. IEEE*, vol. 85, Mar 1997.

[3] G. De Micheli and M. Sami, Eds., *Hardware/Software Co-Design*. ser. E: Applied Sciences. New York: Kluwer, 1996, vol. 310.

[4] P. Koopman, "Embedded systems design issues (the rest of the story)," in *Proc. IEEE ICCD*, 1996.

[5] C. Alippi and L. Briozzo, "Accuracy vs. precision in digital VLSI architectures for signal processing," *IEEE Trans. Comput.*, vol. 47, pp. 472–477, Apr. 1998.

[6] S. Piché, "The selection of weights accuracies for Madalines," *IEEE Trans. Neural Networks*, vol. 6, pp. 432–445, Mar. 1995.

[7] M. Stevenson, R. Winter, and B. Widrow, "Sensitivity of feedforward neural networks to weights errors," *IEEE Trans. Neural Networks*, vol. 1, pp. 71–80, Mar. 1990.

[8] J. Holt and J. Hwang, "Finite precision error analysis of neural network hardware implementations," *IEEE Trans. Comput.*, vol. 42, pp. 281–290, Mar. 1993.

[9] G. Dundar and K. Rose, "The effects of quantization on multilayer neural networks," *IEEE Trans. Neural Networks*, vol. 6, pp. 1446–1451, Nov. 1995.

[10] C. Alippi, V. Piuri, and M. Sami, "Sensitivity to errors in artificial neural networks: A behavioral approach," *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 358–361, June 1995.

[11] J. Dugundji, *Topology*. Boston, MA: Allyn and Bacon, 1966.

[12] H. Taub and D. L. Schilling, *Principles of Communications Systems*. New York: McGraw-Hill, 1986.

[13] M. J. Buckingham, *Noise in Electronic Devices and Systems*. Horwood, U.K.: Chichester, 1983.

[14] B. R. Barmish and C. M. Lagoa, "The uniform distribution:A rigorous justification for its use in robustness analysis," *Math. Control, Signals Syst.*, vol. 10, pp. 203–222, 1997.

[15] E. Bai, R. Tempo, and M. Fu, "Worst-case properties of the uniform distribution and randomized algorithms for robustness analysis," in *Proc. IEEE Amer. Control Conf.*, Albuquerque, NM, 1997, pp. 861–865.

[16] C. Alippi, "Randomized algorithms: A system level, poly-time analysis of robust computation," IEEE Trans. Comput., submitted for publication.

[17] S. P. Bhattacharyya, H. Chapellat, and L. H. Keel, *Robust Control: The Parametric Approach*. Englewoods Cliffs, NJ: Prentice-Hall, 1995.

[18] L. R. Ray and R. F. Stengel, "A Monte Carlo approach to the analysis of control system robustness," *Automatica*, vol. 29, pp. 229–236.

[19] R. Tempo and F. Dabbene, "Probabilistic robustness analysis and design of uncertain systems," *Progress in Syst. Control Theory*, vol. 25, pp. 263–282, 1999.

[20] L. Ljung, *System Identification, Theory for the User*. Englewoods Cliffs, NJ: Prentice-Hall, 1987.

[21] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. New York: Addison-Wesley, 1991.

[22] M. H. Hassoun, *Fundametals of Artificial Neural Networks*. Cambridge, MA: MIT Press, 1995.

[23] Y. Mallet, D. Coomans, J. Kautsky, and O. De Vel, "Classification using adaptive wavelets for feature extraction," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 1058–1066, Oct. 1997.

[24] H. Dawid and H. Meyr, "The differential CORDIC algorithm: Constant scale factor redundant implementation without correcting iterations," *IEEE Trans. Comput.*, vol. 45, pp. 307–318, Mar. 1996.

[25] Y. Hen Hu and H. H. M. Chern, "A novel implementation of CORDIC algorithm using backward angle recoding (BAR)," *IEEE Trans. Comput.*, vol. 45, pp. 1370–1378, Dec. 1996.

[26] A. M. Mood, F. A. Graybill, and D. C. Boes, *Introduction to the Theory of Statistics*, 3rd ed. New York: McGraw-Hill, 1974.

[27] M. Vidyasagar, "An overview of computational learning theory and its applications to neural network training," in *Identification, Adaptation, Learning*, ser. NATO ASI Series F. New York: Springer-Verlag, 1996, vol. 153, pp. 400–422.

[28] ——, "Statistical learning theory and randomized algorithms for control," *IEEE Contr. Syst. Mag.*, pp. 69–85, Dec. 1998.

[29] ——, *A Theory of Learning and Generalization with Applications to Neural Networks and Control Systems*. Berlin, Germany: Springer-Verlag, 1996.

[30] G. Calafiore, F. Dabbene, and R. Tempo, "Uniform sample generation of vectors in l_p balls for probabilistic robustness analysis," in *In Recent Advances in Control*. New York: Springer-Verlag, 1999.

[31] X. Chen and K. Zhou, "On the probabilistic characterization of model uncertainty and robustness," in *Proc. IEEE 36th Conf. Decision and Control*, San Diego, CA, 1997, pp. 3816–3821.

[32] P. Djavdan, H. Tulleken, M. Voetter, H. Verbruggen, and G. Olsder, "Probabilistic Robust Controller Design," in *Proc. IEEE 28th Conf. Decision and Control*, Tampa, FL, 1989, pp. 2144–2172.

[33] Š. Raudys, "On dimensionality, sample size, and classification error of nonparametric linear classification algorithms," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 667–671, June 1997.

[34] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Ann. Math. Stat.*, vol. 23, pp. 493–507, 1952.

[35] B. R. Barmish, *New Tools for Robustness of Linear Systems*. New York: McMillan, 1994.

[36] G. Strang and T. Nguyen, *Wavelet and Filter Banks*: Wellesley-Cambridge, 1996.

[37] S. Basu and B. Levy, *Multidimensional Filter Banks and Wavelet: Basic Theory and Cosine Modulated Filter Banks*. New York: Kluwer, 1996.

Cesare Alippi (S'92–M'97–SM'99) received the Dr. Ing. degree in electronic engineering (*summa cum laude*) in 1990 and the Ph.D. degree in computer engineering, in 1995, both from Politecnico di Milano, Milano, Italy. Further education includes research work in computer sciences carried out at the University College London and the Massachussets Institute of Technology.

Currently, he is an Associate Professor in Information Processing Systems at the Politecnico di Milano. His interests include neural networks (learning theories, implementation issues, and applications), composite systems and high-level analysis and design methodologies for embedded systems. His research results have been published in more that 80 technical papers in international journals and conference proceedings.