Fast Kriging-based Error Evaluation for Approximate Computing Systems

Justine Bonnot, Daniel Menard, Karol Desnos
Univ Rennes, INSA Rennes, IETR UMR 6164, Rennes, France
jbonnot@insa-rennes.fr

***Abstract*—Approximate computing techniques trade-off the performance of an application for its accuracy. The challenge when implementing approximate computing in an application is to efficiently evaluate the quality at the output of the application to optimize the noise budgeting of the different approximation sources. It is commonly achieved with an optimization algorithm to minimize the implementation cost of the application subject to a quality constraint. During the optimization process, numerous approximation configurations are tested, and the quality at the output of the application is measured for each configuration with simulations. The optimization process is a time-consuming task. We propose a new method for infering the accuracy or quality metric at the output of an application using kriging, a geostatistical method.**

I. Introduction

The fierce competition to design faster, cheaper and more energy-efficient electronic systems has led to the development of many methods to minimize silicon area, energy consumption and latency. During the design process of a System on Chip (SoC), every choice is important to be able to embed massive applications as in signal, image or video processing, and artificial intelligence fields. To optimize criteria as energy, latency and area, a new approach is to trade-off the output application quality for the cost of the designed system. In this context, numerous approaches have been proposed in Approximate Computing (AC). The approximation technique can be applied at the computation level, by skipping or approximating some processing [1], at the hardware level with voltage overscaling [2] or inexact operators [3]–[5] or at the data level with finite precision arithmetic [6], [7]. Each approximation technique may be tuned according to different parameters to trade quality for performance. Consequently, when implementing AC techniques in an application, the AC design space has to be explored to obtain the best solution minimizing the implementation cost. This Design Space Exploration (DSE) can be modeled as a combinatorial optimization problem searching for the optimal solution in a N_v-dimension hypercube, where N_v is the number of variables to optimize.

Nevertheless, solving this optimization problem is long and complex. Quality evaluation has been identified as one of the most critical process [8], [9]. The quality metric λ is evaluated numerous times during the optimization process. The evaluation must be accurate to take the right decision during the travel inside the N_v-dimension hypercube. The evaluation time must be low enough to obtain reasonable optimization time. Currently, two types of state-of-the-art approaches can be used to evaluate the quality at the output of an application: analytical and simulation-based approaches. Analytical methods mathematically express the quality metric depending on the approximation sources. Approaches based on interval arithmetic have been used to determine the output error bounds [10] or the computation significance [11]. Generalizing theses approaches to other quality metric or to other approximation sources is still a challenge. Moreover, analytical techniques are complex, hard to automate and their applicability can be limited to systems having specific properties. Simulation-based techniques are widely used since they are generic and easier to implement than analytical techniques. The application is simulated on an arbitrary large pre-defined input data set using a reference simulation without approximation and a simulation integrating the approximations. Nevertheless these approaches suffer from a major drawback which is the long evaluation time. Indeed, the approximation technique has to be emulated to measure its impact on the considered quality or accuracy metric. The simulation time overhead due to the approximation emulation combined with the great number of input values to simulate and the numerous configurations to test lead to long evaluation time to evaluate the metric at the output of the application. Different approaches have been proposed to reduce the overhead due to the emulation of approximation techniques [12], [13] or to reduce the number of input values to simulate [14].

In this paper, we propose a new approach based on statistical interpolation to reduce the number of simulation-based metric evaluations. The concept exploited in this paper is kriging, a geostatistical inference method compatible with non-linear systems and any type of accuracy or quality metric. Kriging allows interpolating the value of the metric λ in a given configuration, from previously evaluated values of λ in other configurations. In this paper, our method is illustrated with two types of optimization problem for the AC DSE. The first one is the case of finite precision refinement, where the variables to optimize are the application data word-lengths. The second one is the case of error sensitivity analysis. The aim of this optimization process is to find the maximal tolerated power of internal error sources for a targeted quality metric value. This approach can be used for other AC DSE as long as the interpolated surface is continuous and a distance between the different tested configurations can be defined.

The remainder of this paper is organized as follows: Section II presents the context and related works to solve the considered optimization problem. Section III details the proposed interpolation-based method. Section IV presents the experimental setup and the obtained results in terms of quality of the result of the optimization problem and number of simulated configurations.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732105 (CERBERO project) and from the French Agence Nationale de la Recherche under grant ANR-15-CE25-0015 (ARTEFaCT project)

II. Context and Related works

The AC DSE can be modeled as an optimization problem in which the implementation cost $\mathcal{C}$ is minimized subject to a quality or accuracy constraint $\lambda_{\min}$:

$$\min(\mathcal{C}(\mathbf{e})) \quad \text{subject to} \quad \lambda(\mathbf{e}) > \lambda_{min} \tag{1}$$

where $\mathbf{e}$ represents a N_v-length vector of the different approximation sources. The value of the metric $\lambda(\mathbf{e})$ depends on the considered configuration of the approximation sources $\mathbf{e}$. In the rest of the paper, the vector of size N_v, $\mathbf{e^i} = (e_0^i, e_1^i, ...e_{N_v}^i)$ represents the configuration i of the different approximation sources. To solve the combinatorial optimization problem, numerous configurations are tested and this leads to long simulation time. The total simulation time t_{opt} to solve the optimization problem depicted in Equation 1 is expressed as:

$$t_{\text{opt}} = N_\lambda \cdot N_o \cdot t_o \tag{2}$$

where N_λ is the number of quality or accuracy metric evaluations, N_o, the number of observations required to accurately estimate the metric λ and t_o, the simulation time of a single observation.

To limit the number of tested configurations, heuristics based on greedy algorithms are commonly used. For instance, for finite-precision refinement, different gradient-based algorithms [15]–[17] using sensitivity information have been proposed. Likewise, in this domain, the simulation time t_o has been reduced in the literature with efficient emulation methods, reducing the time t_o, as for instance C++ fixed-point libraries proposed by Mentor Graphics [12] or by SystemC [13]. A framework based on inferential statistics has also been proposed [14] to reduce the number of observations N_o required to evaluate an intermediate accuracy metric, the noise power.

Complementary to these approaches, in this paper, we focus on interpolation-based techniques to evaluate the quality metric λ. Interpolation-based methods have already been proposed to solve the optimization problem depicted in Equation 1 for fixed-point refinement. In this case, the vector of the different approximation sources is $\mathbf{w}$, the word-lengths vector of the internal variables in the application. Sedano et al. [18] have proposed an interpolation method based on an optimization algorithm similar to the *min+1 bit* algorithm described in Section III-B.

Interpolation is only used during the first step of the considered heuristic for which only the contribution of a single variable on the metric is considered. This approach does not consider a N_v-dimension hypercube allowing taking into account the contributions of all the variables at the same time. Besides, this method is specific to the estimation of the considered intermediate accuracy metric, the noise power. Contrary to [18], our method reduces N_λ using kriging, which is compatible with non-linear systems and any type of accuracy or quality metric. Kriging allows interpolating the value of the metric λ in a given configuration $\mathbf{e^i}$, from previously measured values of λ in other configurations.

III. Proposed kriging-based error evaluation method

The objectives of the proposed method are: 1) to reduce the number of simulation-based evaluations N_λ of the accuracy or quality metric at the output of the application for solving the optimization problem in Equation 1. 2) To provide an interpolation method taking into account a N_v-dimension hypercube. 3) To provide an interpolation method generic for any metric at the output of an application.

If the components of vector $\mathbf{e}$ are forming a hypercube, the different vector elements e_k for each tested configuration are sampling this N_v-dimension hypercube. The proposed method infers the value of the metric λ in a new configuration of the hypercube $\mathbf{e^i}$ from the values of the metric λ already measured on the other configurations, instead of evaluating by simulations. The inference is done with kriging, a technique to estimate the value of a random field, in this case λ, in an arbitrary configuration $\mathbf{e^i}$ depending on the values of λ already measured in the configurations $\mathbf{e^j}$, $j \neq i$.

A. Kriging description

Geostatistics [19] applies the theory of random functions to spatially distributed data. The goal of geostatistics is to model the behavior of a variable that is evolving in space and/or time, and predict its value in unknown parts of space. Geostatistical methods were first developed for mining, and have been used to estimate complex quantities such as confidence intervals. The geostatistical method implemented to estimate the metric is a simple kriging technique. Kriging is a stochastic spatial interpolation technique, that allows predicting a random field $\lambda(.)$, possibly non-linear, in an arbitrary configuration $\mathbf{e^i}$ using the already known values of $\lambda(.)$ in the configurations $\mathbf{e^0}, ..., \mathbf{e^{i-1}}$. The proposed method relies on two steps. Firstly, the function indicating the correlation between points depending on their distance is identified from the already known values of $\lambda(.)$ in $\mathbf{e^0}, ..., \mathbf{e^{i-1}}$. Secondly, the obtained model is used to interpolate the value of $\lambda(.)$ in configuration $\mathbf{e^i}$.

The already known values of λ in configurations $\mathbf{e^0}, ..., \mathbf{e^{i-1}}$ that are used for the interpolation can be discrete or continuous. The interpolated surface does not need to be regularly sampled. Nevertheless, the hypothesis of the proposed method lies in the fact that the interpolated surface is continuous.

The random field $\lambda(.)$ in configuration $\mathbf{e^i}$ is modeled by:

$$\lambda(\mathbf{e^i}) = m + \sum_{k=0}^{i-1} \mu_k \lambda(\mathbf{e^k}) \tag{3}$$

where m and μ_k are constant values. The weights μ_k are determined such that the estimator $\lambda(\mathbf{e^i})$ is unbiased and leads to an estimation error of minimal standard deviation.

The first step of the proposed method consists in deriving the function indicating the evolution of the correlation between the measured values of the random field $\lambda(.)$ in configurations $\mathbf{e^j}$, $j \neq i$, depending on the distance d between the configurations. This function is called the semi-variogram $\hat{\gamma}$. The computation of $\hat{\gamma}(d)$ is detailed in Equation 4.

$$\hat{\gamma}(d) = \frac{1}{2|N(d)|} \sum_{N(d)} \{\lambda(\mathbf{e^j}) - \lambda(\mathbf{e^k})\}^2 \tag{4}$$

where the set $N(d) = \{(j,k) \text{ such that } |\mathbf{e^j} - \mathbf{e^k} = d|\}$ and

$|N(d)|$ represents the number of distinct couples (j,k) in the set $N(d)$. $\mathbf{e^i}$ is the configuration in which the value of λ has to be infered. From the already measured values of $\lambda(.)$, the semi-variogram can be computed and identified to a particular type of semi-variogram [19]. This identification allows computing the value $\hat{\gamma}(d)$ for any value of d. The identification of the semi-variogram has to be done once for a particular metric and application.

Kriging is an optimal linear estimator with no bias. The interpolated value of the metric in configuration $\mathbf{e^i}$ is noted $\hat{\lambda}(\mathbf{e^i})$ and the real value $\lambda(\mathbf{e^i})$. Kriging gives the interpolated value by computing the weighted average of the available configurations leading to an estimation error of minimal standard deviation as presented in Equation 5.

$$\min(Var[\hat{\lambda}(\mathbf{e^i}) - \lambda(\mathbf{e^i})]) \tag{5}$$

The methodology for computing the unknown value can be summarized in the three following steps. Firstly, the unknown value $\hat{\lambda}(\mathbf{e^i})$ is modeled as a linear combination of the known values as expressed in Equation 3. Secondly, the universality constraint, which indicates that kriging is an unbiased estimator is expressed as in Equation 6.

$$E[\hat{\lambda}(\mathbf{e^i}) - \lambda(\mathbf{e^i})] = 0 \tag{6}$$

Thirdly, the optimality constraint is defined by solving Equation 5. The conditions for kriging (optimality and no bias) allow computing the interpolated value $\hat{\lambda}(\mathbf{e^i})$.

Let $\hat{\gamma}_{jk}$ be the semi-variogram value $\hat{\gamma}(|\mathbf{e^j} - \mathbf{e^k}|)$, where $\mathbf{e^j}$ and $\mathbf{e^k}$ are samples in which the value of λ has been measured. Let $\hat{\gamma}_{ik}$ be $\hat{\gamma}(|\mathbf{e^i} - \mathbf{e^k}|)$, where $\mathbf{e^i}$ is the configuration in which the value of λ has to be infered. For clarity, let's denote $\lambda_k = \lambda(\mathbf{e^k})$. If we define two $N+1$-length vectors $\boldsymbol{\lambda}$ and $\boldsymbol{\gamma_i}$ as:

$$\boldsymbol{\lambda} = \quad (\lambda_0, \lambda_1, ..., \lambda_{N-1}, 0) \tag{7}$$

$$\boldsymbol{\gamma_i} = \quad (\hat{\gamma}_{i\,1}, \hat{\gamma}_{i\,2}, ..., \hat{\gamma}_{i\,N-1}, 1) \tag{8}$$

Let the $(N+1) \times (N+1)$ symmetric matrix $\boldsymbol{\Gamma}$ be:

$$\boldsymbol{\Gamma} = \begin{pmatrix} \hat{\gamma}_{0\,0} & \hat{\gamma}_{0\,1} & ... & \hat{\gamma}_{0\,N-1} & 1 \\ \hat{\gamma}_{1\,0} & \hat{\gamma}_{1\,1} & ... & \hat{\gamma}_{1\,N-1} & 1 \\ & & ... & & \\ \hat{\gamma}_{N-1\,0} & \hat{\gamma}_{N-1\,1} & ... & \hat{\gamma}_{N-1\,N-1} & 1 \\ 1 & 1 & ... & 1 & 0 \end{pmatrix} \tag{9}$$

Then, the interpolated value $\hat{\lambda}(\mathbf{e^i})$ is computed as in Equation 10.

$$\hat{\lambda}(\mathbf{e^i}) = \boldsymbol{\gamma_i} \cdot \boldsymbol{\Gamma}^{-1} \cdot \boldsymbol{\lambda} \tag{10}$$

B. Exploitation in the context of an optimization algorithm

As explained in Section II, in the optimization process described in Equation 1, the convergence towards the solution is not done with an exhaustive evaluation of the different configurations but with a optimization algorithm evaluating only a subset of configurations. In the case of greedy algorithm, a local search is carried-out to find the best trajectory in the N_v-dimension hypercube search space. This local search only evaluates a subset of configurations $\mathcal{S}_{ls}$ in the global search space. To evaluate the number of configurations in $\mathcal{S}_{ls}$ for which kriging could be used to evaluate λ, the proposed method has been integrated in a gradient-based greedy optimization algorithm. This particular optimization algorithm can be a steepest descent gradient-based algorithm or a middle ascent gradient-based algorithm. Our method is illustrated on a middle ascent optimization algorithm corresponding to the *min+1 bit* [15] algorithm in the context of word-length optimization.

The challenge is to determine whether the set of already simulated configurations in $\mathcal{S}_{ls}$ allows predicting a large number of configurations using kriging. The optimization algorithm has then been launched on the exhaustive input data set $\mathcal{I}$ to get the real metric values for each tested configuration. For each tested configuration, the word-lengths $\mathbf{w^i}$ of all the variables in the application are recorded as well as the real metric value. The different vectors $\mathbf{w^i}$ are corresponding to the different configurations $\mathbf{e^i}$. Consequently, for each vector of size N_v $\mathbf{w^i} = (w_0^i, w_1^i, ..., w_{N_v}^i)$, the accuracy $\lambda_i = -P_i$ corresponding to the opposite of the real noise power value is measured. The points have been recorded in the order in which they have to be measured, for comparison with the results obtained by kriging. When the noise power value is obtained with simulations, the application is simulated with the considered word-lengths vector $\mathbf{w^i}$ on the exhaustive input data set $\mathcal{I}$ and the accuracy at the output of the application is measured. In the rest of the paper, the simulation of the word-lengths configuration $\mathbf{w^i}$ and the computation of the accuracy metric by simulation are summarized as: $\lambda = \text{evaluateAccuracy}(\mathcal{I}, \mathbf{w^i})$ The goal of the proposed method is to replace the simulation-based metric evaluations by kriging.

1) Proposed algorithm: The proposed method to estimate the metric λ is implemented in the *min+1* bit optimization algorithm described in [15]. For each tested word-lengths configuration $\mathbf{w^i}$, the proposed kriging-based technique has been applied to infer the metric value in configuration $\mathbf{w^i}$ from the surrounding metric values on the hypercube and for a given distance d.

The *min+1* bit optimization algorithm is composed of two steps. The first step determines a minimal word-length vector $\mathbf{w}^{\text{min}}$ used as a starting point for the second step which implements a greedy algorithm to obtained the optimized solution $\mathbf{w}^{\text{res}}$. These two steps have been modified to integrate the proposed krigging-based approach as described in Algorithm 1 for the determination of $\mathbf{w}^{\text{min}}$ and in Algorithm 2 for the determination of $\mathbf{w}^{\text{res}}$. Both algorithms takes as input the following parameters: the accuracy constraint λ_m, the number of variables to optimize N_v and the distance d to search for the neighbours of the interpolated configuration. The impact of parameter d is studied in the experimental study. The value N_{max} corresponds to the maximum tested word-length. The matrix W_{sim} storing the already simulated configuration vectors, the vector storing the corresponding metric values as well as the number of simulated configurations are initialized to the null elements and to zero (line 2). Then, for each tested configuration $\mathbf{w}$, the already simulated configurations are analyzed (lines 7-15) to determine if they can be used for kriging. For each configuration $\mathbf{w}_{sim}^j$ in matrix W_{sim}, its distance to configuration $\mathbf{w}$ in which the metric value is

Algorithm 1 Minimum word-length $\mathbf{w}^{\min}$ determination

1: **procedure** MINKWL($\lambda_m, \mathcal{I}, N_v, N_{max}, d$)
2: $W_{sim} = ()_{0,0}, \lambda_{sim} = ()_{0,1}, N_{sim} = 0$
3: **for** $i \in [1; N_v]$ **do** ▷ Min part
4: $\mathbf{w} \leftarrow (N_{max}, ..., N_{max})$
5: **repeat** ▷ Iterate on the variables
6: $j = 0, W_{tmp} = ()_{0,0}, \lambda_{tmp} = ()_{0,1}, N_n = 0$
7: **while** $j < N_{sim}$ **do** ▷ Iterate on simulated config.
8: $\mathbf{w}^j_{sim} \leftarrow W_{sim}(j, :)$
9: $d_{Cur} = ||\mathbf{w} - \mathbf{w}^j_{sim}||_1$
10: **if** $d_{Cur} \leq d$ **then**
11: $W_{tmp} \leftarrow W_{tmp} \cup \mathbf{w}^j_{sim}$
12: $\lambda_{tmp} \leftarrow \lambda_{tmp} \cup \lambda(j)$
13: $N_n \leftarrow N_n + 1$
14: **end if**
15: $j \leftarrow j + 1$
16: **end while**
17: **if** $N_n > N_{n,min}$ **then** ▷ Process Kriging
18: $\lambda = \text{kriging}(W_{tmp}, \lambda_{tmp}, \mathbf{w})$
19: **else** ▷ Simulation
20: $\lambda = \text{evaluateAccuracy}(\mathcal{I}, \mathbf{w})$
21: $W_{sim} \leftarrow W_{sim} \cup \mathbf{w}$
22: $\lambda_{sim} \leftarrow \lambda_{sim} \cup \lambda$
23: $N_{sim} \leftarrow N_{sim} + 1$
24: **end if**
25: $\mathbf{w}_i \leftarrow \mathbf{w}_i - 1$
26: **until** $\lambda \geq \lambda_m \vee \mathbf{w}_i \leq 1$
27: $\mathbf{w}^{min}_i \leftarrow \mathbf{w}_i + 1$
28: **end for**
29: **return** $\mathbf{w}^{\min}$
30: **end procedure**

Algorithm 2 Optimized word-length $\mathbf{w}^{res}$ determination

1: **procedure** OPTIMKWL($\lambda_m, \mathcal{I}, N_v, \mathbf{w}^{\min}, W_{sim}, \lambda_{sim}, N_{sim}, d$)
2: $\mathbf{w}^{res} \leftarrow \mathbf{w}^{\min}$
3: **repeat**
4: **for** $i \in [1; N_v]$ **do** ▷ Competition between variables
5: $\mathbf{w}_i \leftarrow \mathbf{w}_i + 1$
6: $j = 0, W_{tmp} = ()_{0,0}, \lambda_{tmp} = ()_{0,1}, N_n = 0$
7: **repeat** ▷ Iterate on the simulated configurations
8: $\mathbf{w}^j_{sim} \leftarrow W_{sim}(j, :)$
9: $d_{Cur} = ||\mathbf{w} - \mathbf{w}^j_{sim}||_1$
10: **if** $d_{Cur} \leq d$ **then**
11: $W_{tmp} \leftarrow W_{tmp} \cup \mathbf{w}^j_{sim}$
12: $\lambda_{tmp} \leftarrow \lambda_{tmp} \cup \lambda(j)$
13: $N_n \leftarrow N_n + 1$
14: **end if**
15: $j \leftarrow j + 1$
16: **until** $j < N_{sim}$
17: **if** $N_n > N_{n,min}$ **then** ▷ Process Kriging
18: $\lambda_i = \text{kriging}(W_{tmp}, \lambda_{tmp}, \mathbf{w})$
19: **else** ▷ Simulation
20: $\lambda_i = \text{evaluateAccuracy}(\mathcal{I}, \mathbf{w})$
21: $W_{sim} \leftarrow W_{sim} \cup \mathbf{w}$
22: $\lambda_{sim} \leftarrow \lambda_{sim} \cup \lambda_i$
23: $N_{sim} \leftarrow N_{sim} + 1$
24: **end if**
25: $\mathbf{w} \leftarrow \mathbf{w}^{res}$
26: **end for**
27: $j_c \leftarrow \text{argmin}_i\{\lambda_i\}$
28: $\mathbf{w}^{res}_{j_c} \leftarrow \mathbf{w}^{res}_{j_c} + 1$
29: $\lambda \leftarrow \lambda_{j_c}$
30: **until** $\lambda \leq \lambda_m$
31: **return** $\mathbf{w}^{res}$
32: **end procedure**

searched is computed. The distance is obtained by computing the $\mathcal{L}_1$ norm between both vectors. If the obtained distance is lower or equal to d, the configuration $\mathbf{w}^j_{sim}$ is kept as a neighbouring configuration for kriging. This value is stored in W_{tmp} as well as the corresponding metric value in λ_{tmp} (lines 11-12). If enough surrounding configurations have already been simulated, that is to say if N_n is higher than the minimum number of neighbouring points $N_{n,min}$ (line 17), kriging is applied, else the configuration is simulated. When kriging is applied, from the already measured configurations, the matrix Γ in Equation 9 is computed and the metric value is estimated with Equation 10.

If the configuration is interpolated, it is not used for kriging other configurations. The higher the distance d, the more points can be interpolated.

IV. Experimental study

The experimental study aims at showing that 1) The proposed method can replace simulation for an important number of configurations. 2) The quality of the obtained estimation depends on the number of configurations taken for inference, controlled by the parameters d and $N_{n,min}$. 3) The proposed method can be applied to the estimation of an accuracy or Quality of Service (QoS) metric. The kriging methodology has been implemented with the equations described in [20]. For the estimation of the noise power, an accuracy metric, the proposed method has been applied on the *min+1 bit* algorithm. During this optimization process, numerous word-lengths configurations are tested and their impact on the accuracy metric is measured. For instance, for a Finite Impulse Response (FIR) filter with two variables converted into fixed-point coding, the word-length at the output of the adder and the word-length at the output of the multiplier, the different measurements of the accuracy metric, in this case the noise power, lead to the creation of the surface presented in Figure 1. The goal of the proposed method is to estimate with a sufficient quality a non-negligible number of configurations of the surface without simulations.

In Table I, the obtained results have been reported for the fixed-point refinement of several benchmarks. The distance d is indicated and varies between 2 and 5. For each considered distance, the percentage of configurations that can be interpolated instead of being simulated $p(\%)$ is indicated, as well as the average number of already simulated configurations j that were used for each interpolation. Finally, quality metrics are provided. Let ϵ be the difference between the interpolated and the real value. This difference is expressed as an equivalent number of bits when the accuracy metric is the noise power. In this case, the equivalent number of bits n^i is computed from the noise power value $\hat{P}(\mathbf{w^i})$ in configuration $\mathbf{w^i}$ as: $\hat{P}(\mathbf{w^i}) = \frac{2^{-n^i}}{12}$In this case, ϵ is computed as:

$$\epsilon = \left| \log_2 \left(\frac{\hat{P}(\mathbf{w^i})}{P(\mathbf{w^i})} \right) \right| \tag{11}$$

When the accuracy metric is the noise power, for each interpolated configuration, the equivalent number of bits is computed

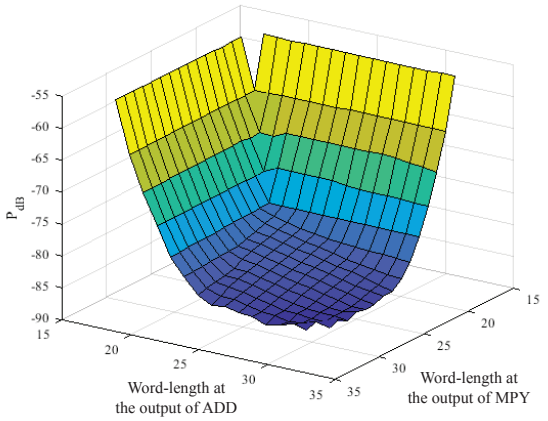

Fig. 1. Evolution of the accuracy metric (noise power in dB) depending on the word-lengths of the adder and the multiplier for a FIR filter.

as well as the equivalent number of bits for the real noise power value. The difference ϵ between the number of bits obtained with the proposed method and with the accurate optimization algorithm is computed and its maximum and average values over all the interpolations are indicated, max_ϵ and μ_ϵ respectively.

When another metric is considered, the difference between the interpolated and the real metric values is expressed as a relative difference. If $\hat{\lambda}(\mathbf{e^i})$ is the interpolated metric value in configuration $\mathbf{e^i}$ and $\lambda(\mathbf{e^i})$ is the accurate metric value, the relative difference is computed as:

$$\epsilon = \frac{|\hat{\lambda}(\mathbf{e^i}) - \lambda(\mathbf{e^i})|}{\lambda(\mathbf{e^i})} \tag{12}$$

The maximum max_ϵ and average μ_ϵ of ϵ are indicated. Among the considered benchmarks, three benchmarks belong to classical signal processing kernels, a 64-th order FIR filter ($N_v = 2$), an 8-th order Infinite Impulse Response (IIR) filter ($N_v = 5$) and a Fast Fourier Transform (FFT) applied on 64 points ($N_v = 10$). For these benchmarks, the chosen metric is the output noise power. The proposed method allows faithfully interpolating a large number of the noise power values from close neighbours. Indeed, when using a distance constraint of $d = 3$ for the FIR filter, 52.78% of the configurations can be interpolated while inducing a low error of interpolation, on average 0.43 bit. For the IIR filter, for $d = 2$, 47.52% of the configurations can be interpolated inducing a similar error of interpolation. With a measured interpolation time of 10^{-6}s compared to a simulation time of 2.4s for the evaluation of a noise power by simulations, the total time for fixed-point refinement is on average halved with our method. It is to be noted that the low number of variables in these two benchmarks leads to a small fraction of the configuration space that can be infered. According to the FFT, the number of variables is larger than for the two filters. As a matter of consequence, a significantly larger number of configurations can be infered from $d = 2$, since 78.14% of the configurations can be infered without using simulations. The error of interpolation also stays really low since it is on average lower than 1 bit for $d \in [\![2; 5]\!]$. In the case of the FFT, if 80% of the noise power evaluations can be done without simulations, the time for quality metric evaluation is divided by 5.

The following considered benchmark is the 2-D motion compensation module of a High Efficiency Video Coding (HEVC) codec. This module processes blocks of 8×8 pixels to interpolate the block in the case of non-integer motion vector. For this module, 23 variables are considered in the word-length optimization process. The considered accuracy metric is the noise power. As for the FFT, since the number of variables in the optimization process is important, a large number of noise power values can be infered, since for $d = 2$, 87.35% of the configurations can be infered. Nevertheless, from $d = 4$, expanding the search space for interpolation is not useful since the percentage of infered configurations is only increasing by 0.35%. The average error of estimation is really low since lower or equal to 0.52 bit and the maximum difference is lower than 2.72 bit. In this case, the inference of the noise power values is really interesting since with a constraint on the noise power of -50dB, 2473 evaluations of the noise power are required. Each evaluation of the motion compensation module by simulation takes 1.37s. Consequently, if 90% of the evaluations can be replaced by interpolation, the time for fixed-point refinement is divided by 10.

The last considered benchmark is a deep learning benchmark with $N_v = 10$ variables. This benchmark is an image classification application based on the SqueezeNet deep convolutional neural network [21]. Contrary to the other tested benchmarks, the optimization problem depicted in Eq. 1 is not a word-length optimization problem but an error sensitivity analysis. An error source is injected at the output of each layer of the network. The configuration $\mathbf{e^i}$ is composed of the power of the different error sources allocated at the output of each layer. The quality metric for this benchmark is the probability p_{cl} to have the same classification as the one predicted by the reference, *i.e.* the classification obtained without error injection. This metric is computed on an input data set composed of 1000 images. The aim of this optimization process is to find the maximal power of the error sources tolerated for a targeted value of p_{cl}. The steepest descent gradient-based greedy algorithm proposed in [22] is used to budget the different error sources between the layers. As shown in Table I, this benchmark leads to very similar results than the FFT in terms of percentage of configurations that can be interpolated without simulations since they have the same number of variables to optimize. The difference between the interpolated and the real value is expressed as a relative difference as presented in Equation 12. The maximum relative difference ranges between 15.72% and 33.58% but is on average lower than 12.16%. For a distance $d = 3$, almost 90% of the configurations can be estimated with the proposed method instead of simulations, while inducing an average relative error of 6.51%. Finally, when solving the optimization problem for the SqueezeNet benchmark with simulations, 290 configurations are tested for an optimization time of 98 hours. If the proposed method is implemented with $d = 3$, 89.31% of the configurations can be estimated with kriging. In this case, the optimization time is divided by a factor 10.

To evaluate the impact of kriging on the result of the optimization algorithm, the number of different decisions (when using kriging), taken during the optimization process has been

measured and approximately ranges 10%. Nevertheless, the optimization algorithm compensates these different choices to end with a similar result than the one obtained without kriging.

To end with, the proposed method has been tested with $N_{n,min} = 2$. Nevertheless, it only reduces the number of configurations that can be interpolated while slightly increasing the interpolation error.

	$\boldsymbol{\lambda}$	N_v	**d**	**p**(%)	**j**	**max**$_\epsilon$	$\boldsymbol{\mu}_\epsilon$
FIR	Noise Power	2	2	33.33	3.78	0.98	0.28
			3	52.78	5.44	1.66	0.43
			4	58.33	7.00	2.29	0.46
			5	66.67	8.61	2.42	0.51
IIR	Noise Power	5	2	47.52	2.72	1.29	0.44
			3	64.54	2.09	2.58	0.72
			4	70.92	2.00	3.24	1.02
			5	77.30	2.00	3.93	1.24
FFT	Noise Power	10	2	78.14	3.48	0.82	0.18
			3	89.07	2.01	1.21	0.34
			4	91.90	2.04	2.07	0.54
			5	95.55	2.05	2.88	0.68
HEVC	Noise Power	23	2	87.35	3.60	1.86	0.07
			3	93.33	2.38	1.86	0.15
			4	95.63	2.11	2.24	0.30
			5	95.96	2.01	2.72	0.52
SqueezeNet	Classification rate	10	2	78.28	3.33	15.72%	3.50%
			3	89.31	2.18	25.75%	6.51%
			4	91.38	2.12	31.57%	9.11%
			5	93.10	2.09	33.58%	12.16%

TABLE I. EXPERIMENTAL RESULTS FOR THE PROPOSED METHOD.

V. CONCLUSION

In this paper, we proposed an interpolation-based method which allows estimating the accuracy or quality metric at the output of an application depending on the different sources of approximation. The estimation is done with a geostatistical method, kriging. The number of estimated configurations without simulations depends on d, the distance between the estimated configuration and its neighbours taken for the estimation. We have verified that with a tight proximity in the neighbouring configurations, kriging enables halving the number of accuracy or quality metric evaluations with simulations, while keeping an estimation error lower than 0.5 bit for small signal processing benchmarks composed of a few variables. When the number of variables in the considered benchmark increases, the search space for the configurations used for kriging is larger and the number of configurations that can be estimated increases up to 90% on average. The proposed method is particularly interesting to solve a multi-dimensional optimization problem and its major advantage is that it is not dependent on a particular metric which is particularly interesting when the chosen metric is hard to evaluate.

REFERENCES

[1] E. Nogues, D. Menard, and M. Pelcat, "Algorithmic-level approximate computing applied to energy efficient hevc decoding," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 1, pp. 5–17, 2016.

[2] S. Lee, L. K. John, and A. Gerstlauer, "High-level synthesis of approximate hardware under joint precision and voltage scaling," in *Proceedings of the Conference on Design, Automation & Test in Europe*, pp. 187–192, European Design and Automation Association, 2017.

[3] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2012.

[4] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proceedings of the 49th Annual Design Automation Conference*, pp. 820–825, ACM, 2012.

[5] V. Mrazek, Z. Vasicek, L. Sekanina, H. Jiang, and J. Han, "Scalable construction of approximate multipliers with formally guaranteed worst case error," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 11, pp. 2572–2576, 2018.

[6] B. Barrois and O. Sentieys, "Customizing fixed-point and floating-point arithmetica case study in k-means clustering," in *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, IEEE, 2017.

[7] A. C. I. Malossi, M. Schaffner, A. Molnos, L. Gammaitoni, G. Tagliavini, A. Emerson, A. Tomás, D. S. Nikolopoulos, E. Flamand, and N. Wehn, "The transprecision computing paradigm: Concept, design, and applications," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1105–1110, IEEE, 2018.

[8] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 62, 2016.

[9] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "MACACO: modeling and analysis of circuits for approximate computing," in *2011 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2011, San Jose, California, USA, November 7-10, 2011*, pp. 667–673, 2011.

[10] C. F. Fang, R. A. Rutenbar, M. Püschel, and T. Chen, "Toward efficient static analysis of finite-precision effects in dsp applications via affine arithmetic modeling," in *Proceedings of the 40th annual Design Automation Conference*, pp. 496–501, ACM, 2003.

[11] V. Vassiliadis, J. Riehme, J. Deussen, K. Parasyris, C. D. Antonopoulos, N. Bellas, S. Lalis, and U. Naumann, "Towards automatic significance analysis for approximate computing," in *Proceedings of the 2016 International Symposium on Code Generation and Optimization, CGO 2016, Barcelona, Spain, March 12-18, 2016*, pp. 182–193, 2016.

[12] M. Graphics, "Algorithmic c data-types," 2016.

[13] T. Grotker, S. Liao, G. Martin, and S. Swan, "System design with systemc," 2010.

[14] J. Bonnot, K. Desnos, and D. Menard, "Accuracy evaluation based on simulation for finite precision systems using inferential statistics," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1508–1512, IEEE, 2019.

[15] M.-A. Cantin and al., "An automatic word length determination method," in *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*, vol. 5, pp. 53–56, IEEE, 2001.

[16] K. Han and B. L. Evans, "Optimum wordlength search using sensitivity information," *EURASIP Journal on Advances in Signal Processing*, vol. 2006, p. 092849, Dec 2006.

[17] H. . Nguyen, D. Menard, and O. Sentieys, "Novel algorithms for word-length optimization," in *2011 19th European Signal Processing Conference*, pp. 1944–1948, Aug 2011.

[18] E. Sedano, J. A. López, and C. Carreras, "A fast interpolative wordlength optimization method for dsp systems," in *2012 VIII Southern Conference on Programmable Logic*, pp. 1–6, IEEE, 2012.

[19] H. Wackernagel, "Geostatistics," *Wiley StatsRef: Statistics Reference Online*, 2014.

[20] B. P. Flannery, W. H. Press, S. A. Teukolsky, and W. Vetterling, "Numerical recipes in c," *Press Syndicate of the University of Cambridge, New York*, vol. 24, p. 78, 1992.

[21] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[22] K. Parashar, R. Rocher, D. Menard, and O. Sentieys, "A hierarchical methodology for word-length optimization of signal processing systems," in *2010 23rd International Conference on VLSI Design*, pp. 318–323, IEEE, 2010.