Circuits, Systems, and Signal Processing (2022) 41:4512–4546
https://doi.org/10.1007/s00034-022-01988-7

Precision Analysis for an Optimal Parallel IIR Filter's Implementation

Mohammed Zelmat[1,2] · El-Sedik Lamini[1] · Samir Tagzout[3] · Hacène Belbachir[1,4] · Adel Belouchrani[5]

Received: 21 December 2020 / Revised: 6 February 2022 / Accepted: 7 February 2022 /
Published online: 26 March 2022

Abstract

This paper addresses the precision analysis of filters in the parallel form framework. The precision analysis consists of determining suitable fractional bit-widths to set a tradeoff between resource consumption and computational accuracy. Although the stability of the parallel form is relatively better controlled than the related direct form, its bit-width optimization did not receive much consideration in the literature, despite its significant contribution to the optimization of the filter's physical implementation. To carry out the needed bit-width optimization, we present two heuristics based on the Estimation of Distribution Algorithm, which falls within the category of probabilistic model-building genetic algorithm. The performance of the proposed approach is discussed, and compared to chosen benchmarks, the results show that our hardware implementations reduce the cost of the resulted circuits up to 37%.

✉ Mohammed Zelmat
mzelmat@usthb.dz

El-Sedik Lamini
elsedik.lamini@usthb.edu.dz

Samir Tagzout
samir.tagzout@proxylan.dz

Hacène Belbachir
hbelbachir@usthb.dz

Adel Belouchrani
adel.belouchrani@g.enp.edu.dz

1 RECITS Laboratory, Department of Operational Research, Faculty of Mathematics, USTHB, Algiers, Algeria

2 Research Center in Applied Economics for Development, Algiers, Algeria

3 EPE PROXYLAN SPA, Lot N0 1, Route de Fouka, Kolea, Tipaza, Algeria

4 Research Centre for Scientific and Technical Information, CERIST, Algiers, Algeria

5 LDCCP Laboratory, Department of Electrical Engineering, Ecole Nationale Polytechnique, El Harrach, Algiers, Algeria

Keywords Bit-width optimization · Estimation of distribution algorithm · IIR filter · Parallel form · Precision analysis

1 Introduction

In recent decades, a considerable research effort has been dedicated to the Infinite Impulse Response (IIR) adaptive filtering studies [15, 31]. Reviewing the literature that deals with the adaptative IIR filters shows that the direct form was often preferred, mainly for its simplicity of implementation and configuration [10, 44]. However, it is often difficult to ensure the stability of the adaptive filter in the direct form. To solve these difficulties, several alternative have been proposed, such as the cascading parallel forms [39]. Parallel structures allow simple monitoring of stability with a smaller complexity than that of the direct ones. However, these advancements also define several new design problems in the choice of bit-widths used in the representation of the variables and the calculated coefficients.

The optimization of digital filter design involves two main steps [41], see Fig. 1 :

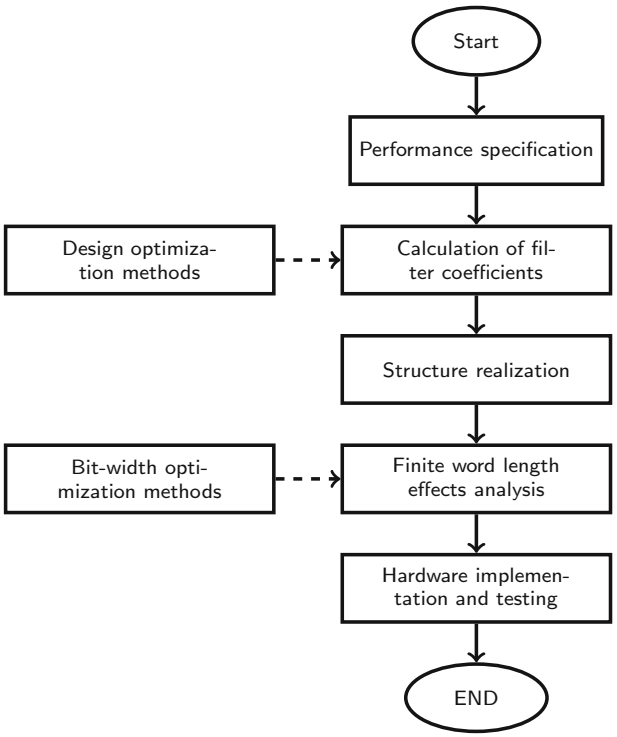

Fig. 1 Flow chart of digital filter design

- Design optimization, which consists of determining the optimal coefficients that guarantee a desired frequency response. Several works have been deployed in order to optimize the design of parallel IIR filters [2, 40];
- Bit-width optimization (BWO), which consists of determining the optimal data widths that ensure computational accuracy while minimizing the hardware implementation's cost [11–13, 18–20, 22, 32, 35–38].

In recent years, several publications have dealt with the implementation and design of the parallel IIR filter. The latter addresses the design of these circuits via the pole placement method to reduce power consumption and numerical noise [3, 17], whereas other authors have proposed new parallel architecture to reduce the filter delay [34]. These contributions do not address the bit-width allocation problem; in most cases, the allocation is done intuitively. Usually, the designer assigns a large width for all the coefficients and interconnections of the filter, following a majorization logic, to avoid any overflow or violation of the precision requirements. Nevertheless, this excessive allocation of bit-widths will waste valuable hardware resources. Moreover, it is well known that a significant portion of the bits is useless or even unused during the program execution process. For this purpose, a bit-width analysis is very important to find the minimum number of bits needed for each variable in the program, which guarantees the correctness of execution and resource-saving.

We contribute through this work to the BWO for filters in parallel form, we assume that the design of the filter is previously established, and we look for a best allocation for the coefficients and interconnections of the filter.

Constantinides and Woeginger proposed a formalization of the BWO problem in [12] and a study of its complexity in [13]. The authors define this optimization problem as follows: "Is there a set of minimum widths that minimizes the objective function while respecting the precision constraints?" and demonstrate that this problem is NP-Hard.

Arithmetics in digital signal processing could be classified according to the format used to encode numbers, fixed-point arithmetic and floating point arithmetic. Recently, BWO for floating-point designs has been studied in many works [8, 25, 26]. On the other hand, many methods, studying fixed-point designs, have been developed. In this paper, the fixed-point implementation of parallel digital filters is considered. The BWO problem for fixed-point designs aims on the one hand to minimize the number of bits of the integer part (Integer Bit (IB)) in order to avoid any overflow and, on the other hand, to minimize bit-width for the fractional part (Fractional Bit (FB)) to reach the accuracy requirements through rang analysis and precision analysis, respectively.

In a digital computational integrated circuit, reducing the bit-width of the data path leads to several advantages including:

- The reduction in the consumed area (the silicon);
- Reducing the number of nodes that change state (toggling nodes) leads to an energy consumption reduction;
- Improving the circuit speed. For example, the response time of an adder is dominated by the carry time. The use of fewer bits makes the adder more efficient. This is even more significant when dealing with multipliers.

The key step in determining the IB of a given signal S is to calculate the absolute value of the reached maximum. When the signal S is an input, the variation interval $[S_{\min}, S_{\max}]$ is known. However, when S is an intermediate signal or an output, this requires a more elaborated approach. A study of the architecture must be made to determine the range of variation in S.

The precision analysis is a problem of higher complexity compared to the rang analysis. It consists of determining the FB_s of the circuit under some requirement on the overall calculation error E_{req}.

Examining the literature enabled us to classify the BWO problem-solving approaches into two categories: dynamic analysis, and analytical analysis.

The dynamic approach is based on simulations [18, 38], which can be impractical, as a large number of input vectors are required to have appreciable results, implying a rather long simulation time.

On the other hand, the analytical approach [22, 32] is based on robust mathematical logic. In some cases, they give overestimated but reliable results (i.e., they ensure the absence of spillover and a predefined calculation precision). The execution time is often acceptable compared with respect to dynamic approaches. Analytical approaches are often the most suitable solution for the design of large circuits (number of elementary operations).

This paper focuses on analytical approaches, which can be generic or specific to a certain architecture. The so-called generic analytical approaches, such as the Interval Arithmetic (AI) and the Affine Arithmetic (AA) [11, 35], have the inconvenience of overestimating the bit-widths; for the AA when non-affine operations occur on the calculation path; as for the AI case it is due to the omission of dependencies between certain variables (signals). In other terms, the use of AI or AA overestimates the variation intervals. This had led to propose specific analytical approaches that depend on the nature of the digital integrated circuit architecture.

In recent years, the BWO problem has been addressed in different applications, notably the evaluation of polynomial functions [7, 23] and signal processing applications [27, 47]. It is clear that recursive circuits such as IIR filters require a specific approach to take into account the complication added by feedback loops. Several authors [19, 20, 36, 37] have proposed specific methods for IIR filters to ensure a better bit-width allocation according to the required error (E_{req}) set beforehand. To the best of our knowledge, no specific BWO method for parallel IIR filters has been proposed. The existing literature has largely addressed the application of BWO methods to direct forms; however, their application to the parallel form remains unexplored with only some mentions of their potential applicability to it as in [19, 20, 36, 37]. Its efficient adaptation to the parallel structure remains a real challenge. Therefore, our contribution intends to fill this gap and propose a process that allows dealing with the BWO methods in parallel form context.

The rest of the paper is organized as follows: we describe the problem in Sect. 2; then, we introduce the basic concepts and definitions in Sect. 3. Section 4 describes briefly the methodologies of precision analysis introduced in [20, 36, 37] as used as a FB allocation process for our approach. Section 5 is devoted to present the proposed resolution approach. In Sect. 6, we compare the results of the experiments to the reference points. A conclusion section ends the paper.

2 Problem Description

The efficient application of BWO methods for parallel forms requires a process that allows a better exploitation of the required error by all the filter blocks. Sarbishei et al. [36] proposed a revision of their BWO approach to make it applicable to other forms of IIR filters. As a parallel form is actually a sum of IIR filter direct forms, it is sufficient to apply the algorithm to all IIR filter direct forms. The parallel filter output error is then given by:

$$e = \sum_{j=1}^{N} e_j = \sum_{j=1}^{N} (y_{\text{fixed}(j)}[n] - y_{(j)}[n]), \tag{1}$$

where N is the number of direct form parallel filters; $y_{\text{fixed}(j)}$ is the fixed-point representation of the jth direct form of the IIR filter and y its reference model. Nevertheless, since the resolution algorithm is applied on the whole architecture at once, the approach proposed in [36] does not allow to give advantage to one block over another in terms of tolerated error. The disadvantage of this approach lies in the fact that the complexity of calculation is different from one block to another, which implies that some blocks require a greater tolerance compared to other blocks to achieve a better allocation of bit sizes.

Furthermore, Lamini et al. [20] proposed the application of the BWO method for each of the blocks independently. The separate treatment of the filter blocks proposed in [20] would be improved if it considers fixing in advance the required error of each block.

In all the efforts aiming at improving the BWO method in [19, 20, 36, 37], the architectural criteria of the related circuits are not deeply and particularly considered. Serial or parallel, sequential or combinatorial, synchronous or asynchronous, distributed or not distributed are criteria that impact inevitably the BWO analysis and solutions. In that sense, we are proposing a novel improvement targeting particularly the parallel topology of the IIR filter.

In this paper, we propose a precision analysis approach to guarantee the best bit-with allocation. This approach consists of two steps. The first step aims to develop a process allowing an optimal required error's exploitation. Unlike the existing literature, we propose a prior sharing of the required error on all the filter blocks using a global optimization method. The second step is based on the adaptation of a BWO method. The application of the latter is done using the output of the previous step (shared error).

To elaborate a process for exploiting the required error, first we express the parallel filter's output error as the sum of the direct filters' output errors (Fig. 2). Then, we suppose that each block is limited by its own required error (E_{req_j}).

The output error expressed by Eq. (1) represents the maximum mismatch, and it does not exceed the required error E_{req}.

$$e \leq E_{\text{req}}. \tag{2}$$

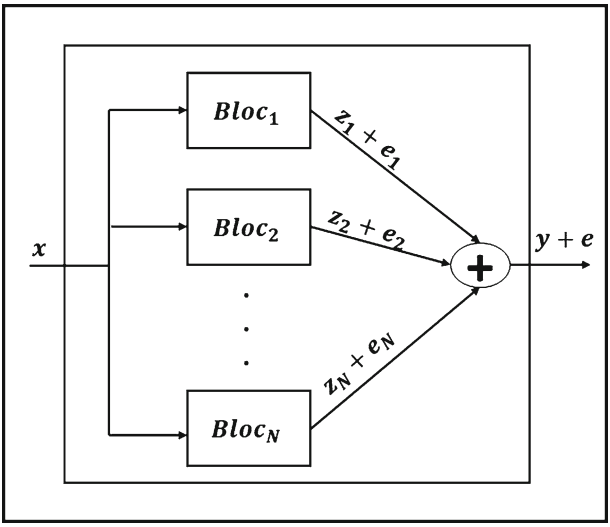

Fig. 2 Parallel filter's output error

Actually, we aim to find the best-required error allocation to provide a good balance between all the direct filters' output errors such that :

$$\sum_{j=1}^{N} E_{\text{req}_j} = E_{\text{req}} \tag{3}$$

and

$$\max(e_j) \leq E_{\text{req}_j}. \tag{4}$$

Once the best error allocation is found, we use one of the BWO methods proposed in [19, 20, 36, 37], in order to determine the best set of IB and FB for all interconnections and coefficients in each block (direct filter). Note that our approach consists of determining the best error allocation which allows bit-width reduction, resulting in an area optimization that is independent of the bit-width allocation process.

The two BWO methods proposed in [36, 37] preserve the stability of the filter, while the BWO method proposed in [20] not only maintains the poles within the unit circle but also preserves the position of the poles to keep the characteristics of the filter.

Given the previous description, we are challenged to find the best distribution of the required error over direct IIR filters' blocks in order to minimize bit-widths of all interconnections of the latter. In the literature, several methods can be adapted to this problem, including probabilistic algorithms. The latter are algorithms that model a problem or search for a problem space using a probabilistic model of candidate solutions. The majority of these algorithms are called Estimation of Distribution Algorithms (EDAs) [29]. EDAs are a family of metaheuristics based on genetic algorithms. In Sect. 5, we clarify the EDAs concept and explain the adaptation of these algorithms for our problem.

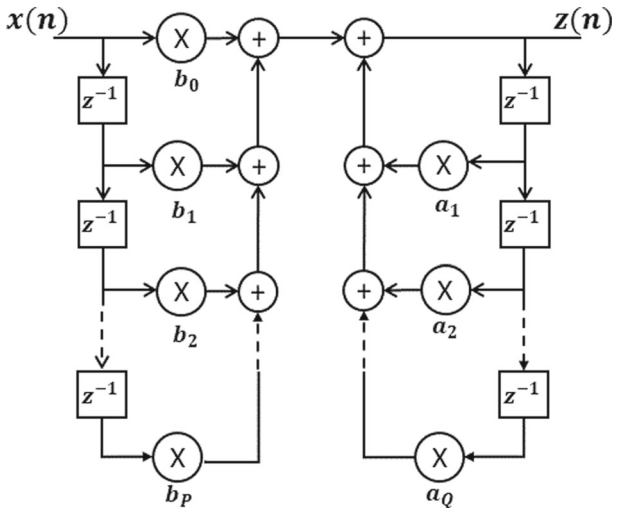

Fig. 3 Direct form structure of IIR Filter

Several works have dealt with the application of genetic algorithms [42, 45] and other metaheuristics such as the Bat algorithm to optimize the design of IIR filters [1], i.e., the determination of optimal coefficients and parameters to meet predefined frequency requirements, while our work consists of optimizing the bit-widths of the coefficients and interconnections of the filter after its design is established.

The ultimate objective is to optimize the cost for the parallel form of IIR filters, in order to reduce its area while respecting the tolerated error bound. Before we detail the proposed solutions, we start by presenting, in the next section, the main basic concepts and definitions related to the problem under consideration.

3 Basic Concepts and Definitions

Digital filters perform filtering operations on incoming samples based on mathematical relationships. Such mathematical operations must be implemented in the hardware in an optimal way. Filters can be represented in several forms among which we may cite the cascade form, the parallel form (Fig. 2), and the direct form (Fig. 3). The actual hardware implementation of a filter is closely linked to the representation used for the latter filter. In general, the matter of choosing an appropriate form can be resolved by the requirements of a particular scenario, since each representation offers advantages in particular situations. The general form of IIR filters [30] usually can be represented by the following formula:

$$z_n = \sum_{i=0}^{P} b_i x_{n-i} + \sum_{j=1}^{Q} a_j z_{n-j}, \tag{5}$$

where x_n is the input signal, z_n is the output signal, P is the feed-forward filter order ($P \leq n$), b_i are the feed-forward filter coefficients, Q is the feedback filter order ($Q \leq n$), and a_i are the feedback filter coefficients.

The z-transform for such an IIR filter is :

$$H[z] = \frac{\sum_{i=0}^{P} b_i z^{-i}}{1 - \sum_{j=1}^{Q} a_j z^{-j}}. \tag{6}$$

3.1 Stability System

The IIR filter is implementable only if the output is bounded for any bound input signal. This reflects the fact that an IIR filter must satisfy the stability condition (Bounded Input Bounded Output (BIBO)). A causal IIR filter is stable if all the poles are inside the unit circle [44].

3.2 Parallel Form of IIR Filters

A parallel structure for the decomposition of a filter into sub-filters is such that the input of each sub-filter is the input of the complete filter and also such that the output of the complete filter is the sum of the outputs of all sub-filters (Fig. 4).

Therefore, to determine such a decomposition amounts to breaking down the transfer function H of the filter into a sum of rational fractions [24].

$$H[z] = \sum_{i=1}^{n} H_i(z). \tag{7}$$

3.3 Fixed-Point Number Representation

Fixed-point calculations are of great importance in signal processing applications. Moreover, floating point calculations, although more accurate, and with a wider dynamic range, require complex hardware implementations and are not as cost effective as fixed-point processing. While the dynamic range of fixed-point numbers can be addressed using more bits in the representation, an optimal design requires using only a sufficient number of bits for the representation of the variables [9]. The total width of the output z is equal to $\mathrm{IB}_z + \mathrm{FB}_z$. IB sets the range, while FB sets the precision. The problem of bit-width assignment is to determine IBs and FBs by rang analysis and precision analysis, respectively.

3.4 Error Model

The hardware implementation of complex mathematical functions in electronic integrated circuits “in ASICs" involves a shift from infinite to finite computational

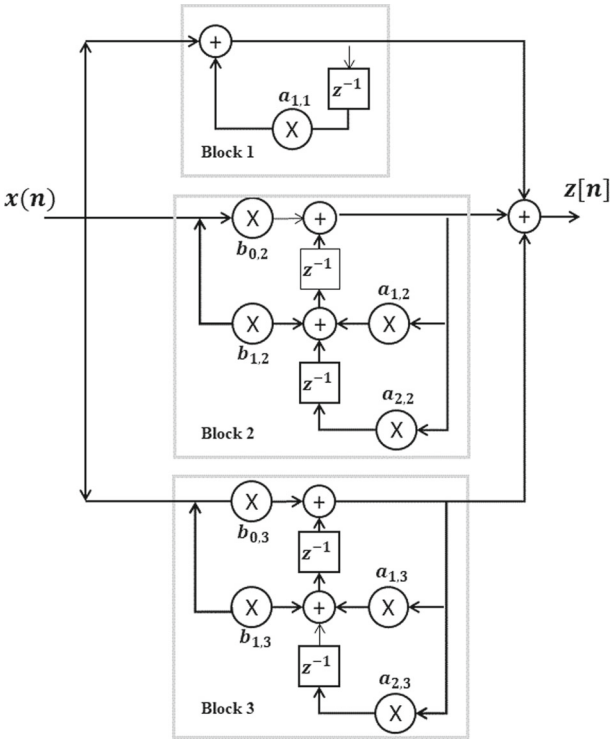

Fig. 4 Parallel form structure for an IIR Filter

accuracy. This transition leads to a loss of precision. This loss is due to the following two reasons:

- The calculation paths using a finite set of elementary operations (addition, subtraction, multiplication, and division) that can be physically implemented. This forces us to give an approximation, based on these elementary operations, for any function we wish to implement physically. This first step generates the approximation error;
- The use of finite bit-widths leads to quantization errors. These propagate in the calculation path and affect the final output results.

Our contribution is related to the quantization error. The error model used in this paper considers the difference between an ideal output value z and its quantized value z_{fxd} and calculated $(\max(|z_n - z_{n_{fxd}}|))$. This difference must be limited by a margin of error such that

$$\lim_{n\to\infty} |z_n - z_{n_{fxd}}| \leq E_{\text{req}}, \tag{8}$$

where E_{req} is an error upper bound when the calculation process tends toward infinity. The accuracy of z is based on the resolution of the floating point computations performed within the processing machine running the program.

3.5 Precision Analysis

Larger FBs allow a more precise calculation. However, larger FBs lead to larger hardware resource conception. The allocation problem of the FBs can be simplified by fixing the same fractional width for all the signals of the circuit Uniform Fractional Bit-width (UFB). Although this solution reduces the complexity of the problem, it leads to an unnecessary consumption of hardware resources. Application-specific integrated circuits (ASICs) allow us to make custom designs (Full-custom) of the targeted application. This means that each signal can have a different data width. This solution is known in the literature as Multiple Fractional Bit-width (MFB). The use of MFBs is potentially capable of leading to much more efficient implementations in terms of hardware cost. The precision analysis consists of three steps:

- Building the error propagation model: This step consists of modeling the error's propagation that is due to the quantification of the input signals and the intermediate signals in the computation path. The objective of this step is to find a general formula for error propagation in the studied calculation path.
- Finding an upper bound for the propagation error: the objective of this step is to propose a bound as close as possible to the error propagation formula. A bound close to the real value implies an optimized FB allocation.
- Estimating the error and allocating FBs: In this step, the objective is to find the optimal MFBs sequence, that is, MFBs that minimize hardware costs while respecting the precision constraint ($\text{err}_{\max} \leq E_{\text{req}}$).

3.6 Hardware Cost Estimation Model

Generally speaking, the hardware cost consists of the consumed surface area of silicium. This can be measured in micro-/nanosquare meters and/or a number of components: logic gate, memories, and pre-designed processor. Usually, the NAND gate is considered when the number of logic gates is evoked.

In our context, we use modeling of the consumed area in order to compare two solutions and achieve a fast optimization process. In this paper, the considered model is similar to the one used by Vakili et al. in [46]. We consider the "full adder" as the elementary operation and the unit of measurement. Therefore, the cost of addition/subtraction and multiplication is given by the following two equations.

$$\begin{cases} \text{cost}(x \pm y) = \max(\text{IB}_x, \text{IB}_y) + \min(\text{FB}_x, \text{FB}_y), \\ \text{cost}(x * y) = (\text{IB}_x + \text{FB}_x)(\text{IB}_y + \text{FB}_y). \end{cases} \tag{9}$$

4 Precision Analysis Heuristics Used for Our Approach

In this section, we briefly present the precision analysis heuristics proposed in [36], [20, 37] that are used in our resolution approaches.

Sarbishei et al. [36] introduced an approach to allocate an appropriate UFB to all input signals as well as intermediate signals of the IIR filter, in order to cope with the constraint on the computational error

$$\max(|\mathrm{err}_z|) \leq E_{\mathrm{req}}. \tag{10}$$

This approach starts with an initial UFB value that does not satisfy the constraint on the computational error. The UFB value is incremented iteratively and the procedure stops when the constraint expressed in Eq. 10 is satisfied. However, reference [36] does not provide a process to calculate the IB_s of the intermediate signals. Their work is limited to the calculation of the UFB value and considers that all the errors of intermediate quantifications are equal. Sarbishei et al. present in [37] a method for calculating MFB.

The authors propose the fixed-point representation of z_n as a function of the quantized impulse response and intermediate quantization errors. They adapt the UFB allocation approach proposed in [36] for MFB allocation in the first step. They allocate the uniform fractional bit for filter coefficients (FBc), while assuming that intermediate quantification errors are zero. The proposed approach initializes FBc to a very small value. The value of FBc is incremented until the error constraint is satisfied.

Lamini et al. [20] present a constructive heuristic for the precision analysis problem. For this purpose, they reformulate the equation that expresses the propagation of errors in an IIR filter. Then, they introduce an upper limit of this equation in order to be able to follow a MFBs allocation process. The proposed process for allocating MFBs in this method is a constructive process of improvement based on several stages. The first step consists of an analytical calculation of the Initial Uniform Fractional Bit-width (IUFB), which is a lower bound for the UFB. The latter is calculated with an incremental search algorithm from the IUFB. In the second step, an adjustment algorithm allocates specific FBc values to each coefficient. In the third step, an effective fractional bit-width of the intermediate variables (FBiv) is calculated by an analytic formula. The final step refines these results by reducing as much as possible the FBz and FBx, the widths of the output signal and the input signal, respectively.

5 The Proposed Resolution Approach

In this section, we describe a resolution approach for our problem. We study the use of two algorithms based on the EDA. EDA, also known as probabilistic model-building genetic algorithms (PMBGA), is an extension of the field of evolutionary computation that models a population of candidate solutions as a probabilistic model. They are used to solve the hard optimization problems through the manipulation of a sampling function describing the quality of possible solutions. Like all metaheuristics using a population of solutions, this method is iterative [21]. In contrast with “classical" evolutionary algorithms, the core of the EDA methods consists of estimating the relations between different variables of an optimization problem by using the estimation of a probability distribution associated with each point of the sample. Therefore, EDA methods do not use crossing or mutation operators. Instead,

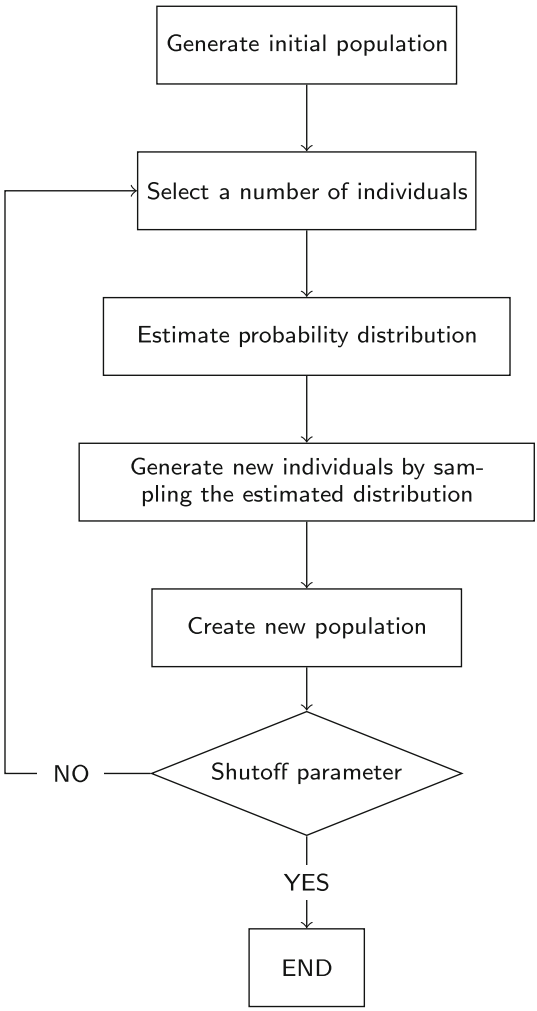

Fig. 5 Description of an Estimation of Distribution Algorithm

the sample is directly constructed from the distribution parameters estimated at the previous iteration.

The basic algorithm proceeds as follows for each iteration:

- Random drawing of a set of points according to a given probability distribution;
- Selection of the best points;
- Extraction of the parameters of the probability distribution describing the distribution of these points.

More precisely, the algorithm proceeds according to the diagram in Fig. 5.

The presented diagram constitutes the structure of the basic algorithm of the EDAs. However, several variants of the EDA have been proposed; each one differs

depending on the chosen probabilistic model. These variants can be classified into three categories:

- Models without dependencies where the probability distribution is constructed from a set of distributions that are defined on a single variable. In other words, the distribution is factorized from univariate, independent distributions on each variable. The most well-known algorithms of non-dependent models are Population-Based Incremental Learning (PBIL) [4] and Univariate Marginal Distribution Algorithm (UMDA) [28];
- Models with bi-variant dependencies in which it is possible to use bi-variant distributions as a basis. Among these algorithms, we may cite Mutual Information Maximizing Input Clustering (MIMIC) Algorithm [14], Combining Optimizers with Mutual Information Tress (COMIT) [6];
- Models with multi-variant dependencies where all dependencies are taken into account in the model. The most well-known algorithms of this category are ECGA (Extended Compact Genetic Algorithm) [16] and BOA (Bayesian Optimization Algorithm) [33].

For our problem, since the individuals of a population are distinct and represent a given partition of the required error over all the blocks of the filter, we have a model without dependencies, which warrants the proposal of two resolution approaches based on the two algorithms UMDA and PBIL to ensure the best way to partition the required error of the filter output over all its blocks. The advantage of this method lies in the technique that allows to locate the most promising areas of the search space, in order to quickly converge toward a better solution.

Next, we briefly present the two algorithms used for our resolution approach followed by the adaptation of the last two to our problem.

5.1 Univariate Marginal Distribution Algorithm

The UMDA has been proposed by Pelikan and Myhlenbein [28]. The information processing strategy of the algorithm consists of using the frequency of the components in a population of candidate solutions in the construction of new candidate solutions. To do so, we must first measure the frequency of each component in the population (the univariate marginal probability) and use the probabilities to influence the probabilistic selection of the components in the component-wise construction of new candidate solutions. The UMDA selects a number S of individuals according to their fitness. Then, based on these individuals, a probabilistic model is constructed. To estimate the marginal probabilities $p(x)$ from a selected population S, by assuming S contain λ elements, this template can be written as follows:

$$p(x) = \frac{1}{\lambda} \sum_{x \in S} x_k, \forall k \in \{1, 2, \ldots, N\}, . \tag{11}$$

Finally, a new solution will be generated based on this model.

5.2 Population-Based Incremental Learning

The PBIL algorithm is originally inspired by competitive learning and is designed for binary problems. It is widely studied in the literature although a particular reference should be made for [4, 5]. It consists of transforming the population of candidate solutions into a probability vector $p(x) = \{p(x_1), p(x_2), \ldots, p(x_n)\}$, where $p(x_i)$ denotes the probability of having a 1 in the ith position of the solution bits ($i = 1, 2, \ldots, n$). The purpose of the PBIL algorithm is to reduce the memory space required by a genetic algorithm. This is done by reducing the population of candidate solutions to a single probability vector from which candidate solutions can be generated and evaluated. Updates and mutation operators are also performed on the probability vector rather than on the generated candidate solutions . All positions are initially equiprobable, that is, all probabilities are initialized to 0.5. After the coding step of the solution, the PBIL algorithm generates P_1 solutions according to the current probability. Then, each component of the probability vector $p(x_i)$ is updated using the following formula:

$$p(x_i) = p(x_i) + \alpha(x_i^{\text{best}} - p(x_i)) \text{ with } \alpha \in [0, 1]. \tag{12}$$

The parameter α is called the learning rate, and x_i^{best} is the ith bit of the best solution already obtained.

5.3 Adaptation of EDA to the Precision Analysis Problem

We adapt both algorithms to ensure the best way to partition the required filter output error over all filter blocks and to minimize the bit-widths allocated for all blocks as well as to reduce the cost of the filter. For both proposed algorithms, we set as a objective the minimization of the fitness function which corresponds to the sum of the cost of each block of the filter:

$$f = \sum_{i=1}^{N} \text{cost}(H(i)), \tag{13}$$

where $H(i)$ is the ith block of the filter and N is the number of filter blocks.

To calculate the cost function of any architecture, we need to use Eq. (9) and to make a summation on all the operations of the circuit.

Next, we proposed two approaches based on the two algorithms mentioned previously (UMDA and PBIL). The steps of the two proposed algorithms are illustrated by the diagram in Fig. 5, where the difference between the two algorithms lies mainly in the steps of the generation of the initial population, the selection of a number of individuals, and the estimation of the probability distribution. The pseudo-codes of our two algorithms, namely Precision Analysis with UMDA (PA_UMDA (Algorithm 1)) and Precision Analysis with PBIL (PA_PBIL (Algorithm 2)), are presented hereafter. In what follows, we give a brief explanation of the steps :

Algorithm 1: Partition of the E_{req} with PA_UMDA algorithm

Input: N, E_{req}, M, S, a, b,$[x_{\min}, x_{\max}]$.
Output: E_{req_j} : Required error for each block.

```
begin
    Randomly draw M individuals which form the first population by Eq. (16);
    l = 0;
    while (shutoff condition is not verified) do
        l = l + 1 Calculate IB and FB for each bloc by one of the methods ([20, 36, 37]);
        Calculate f(i) by Eq. (9);
        Select the S best individuals (with S < M) from the previous population D_{l-1};
        Calculate P_j(x) by Eq. (18);
        Randomly draw M individuals in P_j(x) by Eq. (19);
    return E_{req_j}
```

Algorithm 2: Partition of the E_{req} with PA_PBIL algorithm

Input: N, E_{req}, M, S, a, b,$[x_{\min}, x_{\max}]$.
Output: E_{req_j} : Required error for each block.

```
begin
    Generate an initial probability vector P(x);
    for i := 0 to n do
        P(x_i) = 0, 5;
    while (shutoff condition is not verified) do
        Generate a sample of M individuals based on P(x);
        Calculate IB and FB for each bloc by one of the methods ([20, 36, 37]);
        Calculate f(i) by Eq. (9);
        Select the x_i^best best individual from the previous population D_{l-1};
        Calculate P(x) by Eq. (12);
        Randomly draw M individuals in P(x) by Eq. (19);
    return E_{req_j}
```

5.3.1 Generation of the First Population

An individual in a population represents a configuration of a partition of the required error E_{req} on all filter blocks.

– For the PA_UMDA algorithm: the initial population is a random distribution of E_{req} over all blocks for any individual in the population, such as:

$$\sum_{j=1}^{N} E_{\text{req}_{ij}} = E_{\text{req}}, \text{ for all } i \in \{1, 2, \ldots, M\}, \tag{14}$$

where $E_{\text{req}_{ij}}$ is the required error of the jth block of the ith individual; N and M are the number of blocks and the number of individuals, respectively.

– For the PA_PBIL algorithm: the initial population is generated according to the probability vector. All positions are initially equiprobable, that is, all probabilities are initialized to 0.5.

So, if we set:

$$p_{ij} = \frac{E_{\text{req}_{ij}}}{E_{\text{req}}}, \tag{15}$$

we have $\sum_{j=1}^{n} p_{ij} = 1$ for all $i = 1, \ldots, M$. The set that represents an individual for both algorithms is introduced as follows:

$$I_n = \{(p_1, \ldots, p_n) : p_j \geq 0, \sum_{j=1}^{n} p_j = 1\}. \tag{16}$$

The same process is repeated for the construction of M individual of the first population D_0.

5.3.2 Evaluation of Solutions

The previous step delivers a population of M individuals, where each individual corresponds to a random distribution of the maximum error over all the blocks. To evaluate each individual:

- We calculate the necessary bit-widths for each block, using the Bit-Width Optimization methods presented in [20, 36, 37];
- We calculate the cost of the filter, using the method introduced in [46].

The fitness function $f(i)$ corresponds to the cost of the filter for each individual. One individual is said to be better than another, if its fitness function is lower than that of the other individual.

5.3.3 Selection of the Best Individuals

After calculating the fitness function $f(i)$ for all individuals:

- For the PA_UMDA algorithm: we select S best individuals according to their fitness function (such as $S < M$) and delete the remaining $M - S$ individuals, to form the population D_{l-1}^{s}. We choose an S that varies between [20–80]% of the population size M. The choice of this range of values is based on simulations during which we noticed that it allowed more solutions to be explored. In fact, the choice of a tighter interval could ignore more promising parts of the research space;
- For the PA_PBIL algorithm: we select x_i^{best} the best individual according to the fitness function.

5.3.4 Estimation of the Probability Distribution

The probabilistic model is the main advantage of EDAs. Also, the performance of the algorithm is closely linked to the choice of this model. This step consists of building

a probabilistic model for the subset of selected individuals. In an attempt to generate a new individual, the EDA builds a probabilistic model from the individuals selected in the previous step. We propose the construction of a probabilistic model generating a set of partitions that constitutes the new generation

– For the PA_UMDA algorithm: we assume that there is no dependency between the individuals of a population. Therefore, the joint probability distribution with N dimensions is defined by the ratio of the partition sum of each block for all individuals over the sum of the total partitions of the S matrix of the best selected individuals. The distribution describing the population D_{l-1}^{s} is defined by :

$$P_j(x) = \frac{\sum_{i=1}^{S} p_{ij}}{\sum_{j=1}^{N}\sum_{i=1}^{M} p_{ij}}, \tag{17}$$

where $j = 1, \ldots, N$ (number of blocs), and $i = 1, \ldots, M$ (number of individuals).
We have $\sum_{j=1}^{N} p_{ij} = 1$, so

$$P_j(x) = \frac{1}{S}\sum_{i=1}^{S} p_{ij}, \tag{18}$$

where p_{ij} is a portion of E_{req} allocated for the jth block of the ith individual.
– For the PA_PBIL algorithm: The basic idea of our model is that it depends on the structure of the best selected partition. The probability vector maintained by PBIL can be viewed as a prototype vector for generating solution vectors which have high evaluations with respect to the available knowledge of the function space. In each generation, the probability vector is adjusted by the learning rate to represent the current highest evaluation vector.
The probability vector is updated using the following formula defined in Eq. (12). Low learning rates are preferred, such as 0.1.

5.3.5 Creation of the New Generation

This step consists of randomly drawing M individuals according to the estimate of the probability distribution $P_j(x)$ while respecting the description of the individual described previously. This amount is obtained by solving the following equation system:

$$\begin{cases} \sum_{j=1}^{N} p_{ij} = 1, \ \forall i = 1, \ldots, M, \\ \sum_{i=1}^{M} p_{ij} = P_j(x), \ \forall j = 1, \ldots, N. \end{cases} \tag{19}$$

The resolution of this system actually consists of finding a matrix which has the sum of the entries of each row equal 1 and the sum of the entries of the jth column equal $P_j(x)$.

5.3.6 Shutoff Condition

The previous steps of the algorithm are repeated in order to improve the individuals of the population at each iteration, until one of the shutoff conditions is satisfied:

- $f(a) = f(b), \forall (a, b) \in I_n$, where I_n is the set of individuals;
- The best of $(f(i))$ appears in k iterations.

The best solution corresponds to the partitions of the required error on the blocks of the filter given by the best individuals at the end of the process.

6 Experimental Results

To highlight the performance of the proposed approaches, we compare our methods with respect to recent works. The comparison is done using the benchmarks proposed in [19, 20, 36, 37]. All proposed algorithms and benchmarks have been implemented with MATLAB. The benchmarks are given below:

Bench#1 (3rd order filter):
Parallel Form: $z[n] = z_1[n] + z_2[n]$,
$z_1[n] = x[n] - 0.2z_1[n-1]$,
$z_2[n] = x[n] - 0.3z_2[n-1] + 0.4z_2[n-2]$.

Bench#2 (3rd order filter):
Parallel Form: $z[n] = z_1[n] + z_2[n]$,
$z_1[n] = 0.24x[n-1] - 0.4z_1[n-1]$,
$z_2[n] = 0.2x[n-1] + 0.25x[n-2] - 0.8z_2[n-1] + 0.5z_2[n-2]$.

Bench#3 (4th order filter):
Parallel Form: $z[n] = z_1[n] + z_2[n]$,
$z_1[n] = 0.11x[n] - 0.1041x[n-1] + 0.11x[n-2] + 1.58z_1[n-1] - 0.6469z_1$
$[n-2]$,
$z_2[n] = 0.2464x[n] - 0.426x[n-1] + 0.2464x[n-2] + 1.7753z_2$
$[n-1] - 0.892z_2[n-2]$.

Bench#4 (5th order filter):
Parallel Form: $z[n] = z_1[n] + z_2[n] + z_3[n]$,
$z_1[n] = 0.13x - 0.267z_1[n-1]$,
$z_2[n] = 0.31x[n] + 0.365x[n-1] - 1.4826z_2[n-1] + 0.827z_2$
$[n-2]$,
$z_3[n] = 0.4926x[n] + 0.286x[n-1] + 1.7452z_3[n-1] - 0.9561$
$z_3[n-2]$.

Bench#5 (6th order filter):
Parallel Form: $z[n] = z_1[n] + z_2[n] + z_3[n]$,
$z_1[n] = x[n] + 1.488x[n-1] + x[n-2] - 1.15z_1[n-1] - 0.451$
$z_1[n-2]$,
$z_2[n] = 0.2375x[n-1] + x[n-2] + 0.6z_2[n-1] + 0.9388z_2[n-2]$,
$z_3[n] = 0.1629x[n] + x[n-2] + 0.79z_3[n-1] - 0.7483z_3[n-2]$.

For the simulations, we worked on a Computer, with the following characteristics:

- Processor: Intel®Core ^{TM}i5-2450M CPU @ 2.50GHz 2.50 GHz
- Random-access memory (RAM): 6.00 GB
- OS: Windows 10, 64-bit

To show the effectiveness of our approaches, we use two types of heuristic analysis precision. The first heuristic provides an appropriate UFB to all input signals and intermediate signals of the IIR filter while the second heuristic offers an improved solution compared to the first by offering a MFB.

First, we apply our approaches using the algorithm proposed in 2010 by Sarbishei et al. [36] as a precision analysis method to calculate the fractional part of all interconnections and filter block coefficients. In that algorithm, the authors proposed a uniform FB for all interconnections and filter coefficients. Then, we compare the results obtained by our approaches with the results presented in [36]. We notice that the proposed algorithms converge toward the same evaluation cost with the advantage of the PA_PBIL algorithm that requires fewer iterations to obtain the best solution for the evaluation cost, knowing that we have set as a shutoff conditions:

- $f(a) = f(b), \forall (a, b) \in I$, where I is the set of individuals;
- If the best of $(f(i))$ appears in k iterations (with $k = 10$).

We notice from our simulations that when the best obtained value of $f(i)$ appears 10 times, this one does not improve even if the number of iterations is increased.

We note that our approaches offer an improvement in all cases. The related improvement reaches 37.31% of the cost. We also note that our proposed algorithms exploit the overall margin of error more efficiently than the method proposed in [36]. This is due to the efficient distribution of the required error over the filter blocks, which means that the proposed approach assigns each block the error value it needs. All the results are displayed in Table 1 with the following inputs:

- Required error : $E_{\text{req}} = 0.1$;
- Input signal: $x \in [-100, 100]$.

For PA_UMDA algorithm:

- Number of individuals in the initial population: $M = 20$;
- The best individuals selected at each iteration : $S = 10$.

For PA_PBIL algorithm:

- Learning rate : $\alpha = 0.1$;
- Number of individuals in the initial population: $M = 20$.

Table 1 Experimental results: comparison to [36]

Bench	$\text{err}_{\max}(E_{\text{req}} = 0.1)$		Evaluation cost (Improvement %)		Iteration	
	[36]	Our approach	[36]	Our approach	PA_UMDA	PA_PBIL
Bench_1	0.0695	0.0977	1464	935 (36.13%)	23	18
Bench_2	0.0754	0.0850	1701	1430 (15.93%)	20	12
Bench_3	0.0272	0.0899	4776	4129 (13.55%)	15	12
Bench_4	0.0869	0.0933	5212	4916 (5.67%)	76	41
Bench_5	0.0777	0.0805	7745	4855 (37.31%)	143	109

Next, we introduce the second type of precision analysis method. In this part, we apply our approach (PA_PBIL) using the algorithm proposed in 2012 by Sarbishei et al. [37] which offers an MFB for the input signals as well as the intermediate signals of the IIR filter. Subsequently, we compare the results obtained by applying the approach proposed in [37] for the parallel structure with the results obtained by applying our approach. Unlike the latter, our approach proposes a different FB for each block of the filter ("/" is used to differentiate between blocks in the different tables); for example, for the first benchmark, the approach presented in [37] proposes an identical FB for all feedback coefficients of all filter blocks ($\text{FB}_a = 14$), while our approach proposes a different FB for the feedback coefficients of each block ($\text{FB}_{a_1} = 10$ and $\text{FB}_{a_2} = 14$).

For this analysis method, the cost reduction reaches 19.88%. For the simulation, we keep the same input variables as in the first experiment. The results are summarized in Table 2 as follows :

- FB_a: Fractional bit-width of coefficients a;
- FB_b: Fractional bit-width of coefficients b;
- FB_{iv}: Fractional bit-width of the intermediate variables;
- FB_x: Fractional bit-width of the input signal;
- FB_z: Fractional bit-width of the output signal.

To visualize the behavior and the performance of the proposed methods, we carry out an analysis while varying the value of the required error of the filter. In this analysis, we use the BWO method proposed in 2018 by Lamini et al. [20] to calculate the integer part (IB) and fractional part (FB) of all interconnections and filter block coefficients. Unlike the first heuristic [36], this method provides an MFB for interconnections and filter coefficients.

We choose this method for its optimized complexity, as it gives an effective solution compared to the other methods proposed in the literature [19, 36, 37]. The robustness, efficiency, and speed of problem resolution with this method are due to the proposed error analysis and the adopted data width allocation process.

First, we propose an equitable partition of the required error on different blocks of the filter and then we compare the results obtained by this partition with the results of the partitions obtained by applying our two approaches. The results of the simulations are shown in Table 3, such as:

Table 2 Experimental results: comparison to [37]

Bench	FBs allocation		Evaluation cost (Improvement %)	
	[37]	Our approach	[37]	Our approach
Bench_1	$\text{FB}_a = 14$	$\text{FB}_a = 10/14$	1008	866 (14.09%)
	$\text{FB}_b = 14$	$\text{FB}_b = 10/14$		
	$\text{FB}_S = 10$	$\text{FB}_{iv} = 10/10$		
	$\text{FB}_x = 7$	$\text{FB}_x = 7/7$		
	$\text{FB}_z = 10$	$\text{FB}_z = 10/10$		
Bench_2	$\text{FB}_a = 12$	$\text{FB}_a = 12/10$	1496	1177 (19.88%)
	$\text{FB}_b = 12$	$\text{FB}_b = 12/10$		
	$\text{FB}_S = 11$	$\text{FB}_{iv} = 9/11$		
	$\text{FB}_x = 9$	$\text{FB}_x = 5/5$		
	$\text{FB}_z = 11$	$\text{FB}_z = 9/11$		
Bench_3	$\text{FB}_a = 20$	$\text{FB}_a = 16/15$	4226	3119 (19.54%)
	$\text{FB}_b = 20$	$\text{FB}_b = 16/15$		
	$\text{FB}_S = 15$	$\text{FB}_{iv} = 13/14$		
	$\text{FB}_x = 13$	$\text{FB}_x = 6/7$		
	$\text{FB}_z = 15$	$\text{FB}_z = 13/14$		
Bench_4	$\text{FB}_a = 18$	$\text{FB}_a = 13/18/17$	4997	4201 (15.93%)
	$\text{FB}_b = 18$	$\text{FB}_b = 13/18/17$		
	$\text{FB}_S = 18$	$\text{FB}_{iv} = 11/14/16$		
	$\text{FB}_x = 14$	$\text{FB}_x = 6/10/10$		
	$\text{FB}_z = 18$	$\text{FB}_z = 11/14/16$		
Bench_5	$\text{FB}_a = 17$	$\text{FB}_a = 14/19/17$	4226	3119 (18.67%)
	$\text{FB}_b = 17$	$\text{FB}_b = 14/19/17$		
	$\text{FB}_S = 17$	$\text{FB}_{iv} = 12/14/12$		
	$\text{FB}_x = 13$	$\text{FB}_x = 8/11/9$		
	$\text{FB}_z = 17$	$\text{FB}_z = 12/14/12$		

- Column 1: presents the five labels of benchmarks used for the simulations;
- Column 2: illustrates the different required error used ($E_{\text{req}} = \{0.9; 0.1; 0.01\}$);
- Column 3: presents the three types of partition proposed for this analysis. First, there is the equitable partition, then a partition according to the PA_UMDA algorithm and finally a partition according to the PA_PBIL algorithm;
- Column 4: is composed of three sub-columns that represent the portion of Ereq allocated for each block. Note that the symbol " $\times$ " is put in the third sub-column for filters that contain only two blocks;
- Column 5: displays the evaluation cost obtained by each of the proposed partitions;
- Column 6: represents the improvement in the evaluation cost of each one of the two proposed algorithms compared to the equitable partition;

Table 3 Experimental results: comparison with the equitable partition [20]

Bench	E_{req}	Partition	Coefficient of partition			Evaluation cost	Improvement (%)	Convergence	
			$Block_1$	$Block_2$	$Block_3$			Time (s)	Iterations
$Bench_1$	0.9	Equitable	0.500	0.500	×	471			
		PA_UMDA	0.3000	0.7000	×	456	3.18%	228.21	15
		PA_PBIL	0.3217	0.6783	×	456	3.18%	173.78	12
	0.1	Equitable	0.500	0.500	×	757			
		PA_UMDA	0.3921	0.6079	×	740	2.24%	271.00	21
		PA_PBIL	0.3964	0.6034	×	740	2.24%	213.14	15
	0.01	Equitable	0.500	0.500	×	1122			
		PA_UMDA	0.3102	0.6088	×	1069	4.72%	254.64	18
		PA_PBIL	0.3093	0.6097	×	1069	4.72%	161.15	14
$Bench_2$	0.9	Equitable	0.500	0.500	×	513			
		PA_UMDA	0.4572	0.5418	×	500	2.53%	415.12	25
		PA_PBIL	0.4520	0.5480	×	500	2.53%	222.84	12
	0.1	Equitable	0.500	0.500	×	894			
		PA_UMDA	0.3108	0.6892	×	832	6.93%	523.34	30
		PA_PBIL	0.3105	0.6895	×	832	6.93%	283.52	14
	0.01	Equitable	0.500	0.500	×	1229			
		PA_UMDA	0.3750	0.6250	×	1198	2.52%	448.60	25
		PA_PBIL	0.3744	0.6256	×	1198	2.52%	228.73	13
$Bench_3$	0.9	Equitable	0.500	0.500	×	2170			
		PA_UMDA	0.4151	0.5849	×	2088	3.78%	739.91	17
		PA_PBIL	0.4058	0.5942	×	2088	3.78%	468.85	11
	0.1	Equitable	0.500	0.500	×	3333			
		PA_UMDA	0.6033	0.3967	×	3119	6.42%	511.83	13
		PA_PBIL	0.6051	0.3949	×	3119	6.42%	492.19	11
	0.01	Equitable	0.500	0.500	×	4477			
		PA_UMDA	0.3881	0.6119	×	4259	4.86%	860.68	20
		PA_PBIL	0.3880	0.6120	×	4259	4.86%	619.70	14
$Bench_4$	0.9	Equitable	0.333	0.333	0.333	2674			
		PA_UMDA	0.0983	0.5206	0.3811	2613	2.28%	680.36	16
		PA_PBIL	0.0984	0.5204	0.3812	2613	2.28%	693.94	15
	0.1	Equitable	0.333	0.333	0.333	4104			
		PA_UMDA	0.0732	0.2541	0.6737	3761	8.35%	93100.11	2067
		PA_PBIL	0.0732	0.2541	0.6737	3761	8.35%	73060.44	1497
	0.01	Equitable	0.333	0.333	0.333	5099			
		PA_UMDA	0.2912	0.3798	0.3290	5057	0.82%	85890.20	1711
		PA_PBIL	0.2920	0.3812	0.3268	5057	0.82%	37850.46	743

Table 3 continued

Bench	E_{req}	Partition	Coefficient of partition			Evaluation cost	Improvement (%)	Convergence	
			$Block_1$	$Block_2$	$Block_3$			Time (s)	Iterations
$Bench_5$	0.9	Equitable	0.333	0.333	0.333	2728			
		PA_UMDA	0.3218	0.2809	0.3937	2521	7.58%	81090.90	2003
		PA_PBIL	0.3225	0.2799	0.3976	2521	7.58%	4950.12	107
	0.1	Equitable	0.333	0.333	0.333	4071			
		PA_UMDA	0.0983	0.5206	0.3811	3669	9.87%	6250.90	147
		PA_PBIL	0.0984	0.5204	0.3812	3669	9.87%	4950.12	107
	0.01	Equitable	0.333	0.333	0.333	5120			
		PA_UMDA	0.0983	0.5206	0.3811	5036	1.11%	6250.90	147
		PA_PBIL	0.0984	0.5204	0.3812	5036	1.11%	4950.12	107

- Column 7: contains two sub-columns. The first represents the required processing time (in seconds) for the two proposed algorithms and the second shows the number of iterations required to achieve the shutoff conditions for the two proposed algorithms;
- Coefficient of partition : The weight of allocated error to each bloc;
- Time (second): The required processing time (second);
- Iterations: The number of iterations required to achieve shutoff condition.

We notice that the proposed algorithms converge toward the same evaluation cost, but not necessarily with the same partition. For example, for $bench_1$ (Ereq $= 0.9$), we notice that the two algorithms provided two dissimilar partitions. PA_UMDA proposed the partition of {0.3000; 0.7000}, while the PA_PBIL algorithm proposed the partition of {0.3217; 0.6783}. On the other hand, for the $bench_4$ (Ereq $= 0.1$), we see that the two algorithms proposed an identical partition {0.732; 0.2541; 0.6737}. The PA_PBIL algorithm seems to perform better in terms of processing time.

We note that our approach offers an improvement in all cases; the improvement reaches 9.87% of the evaluation cost. The results of this analysis are displayed in Table 3, with the following inputs:

- Required error : $E_{req} = \{0.9; 0.1; 0.01\}$;
- Input signal: $x \in [-100, 100]$.

For the PA_UMDA algorithm:

- The number of individuals in the initial population: $M = 50$;
- The best individuals selected at each iteration : $S = 20$.

For the PA_PBIL algorithm:

- The learning rate : $\alpha = 0.1$;
- The number of individuals in the initial population: $M = 50$.

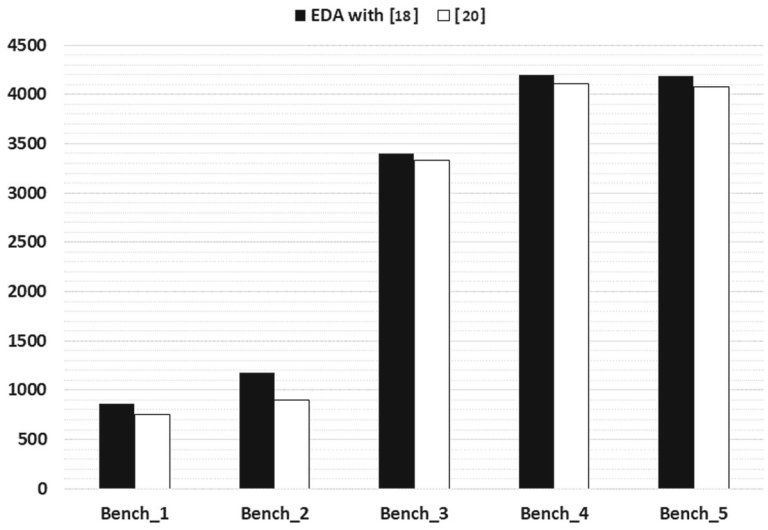

Fig. 6 Evaluation cost

Indeed, by giving the priority to the most expensive blocks, the EDA process manages to induce the above-mentioned improvements.

From the algorithm complexity point of view, the improvement in the evaluation cost is achieved in a reasonable duration. Indeed, the number of populations of almost all the steps of our approach is polynomial functions of their inputs.

Now, we show that our proposed approaches improve the quality of the results regardless of the precision analysis method. In fact, we find that the results obtained by applying our approaches with the use of the precision analysis method presented in [37] are very close to the results obtained with a more recent precision analysis method [20]. All the results are displayed in Fig. 6, with the following inputs:

- Error required: $E_{\text{req}} = 0.01$;
- Input signal: $x \in [-100, 100]$.

Finally, we compare the processing times of the two proposed algorithms. We have established the efficiency of the PA_PBIL algorithm when compared to the PA_UMDA algorithm. Indeed, the high convergence of the former is induced by the reduction in terms of search space which is achieved by the creation of a probability vector. This vector represents a population of best quality solutions where updates and mutation operations are carried out rather than on candidate solutions. This leads to a minimization of the operations' number of the algorithm and therefore of its complexity. We have conducted the last experiment, while varying the parameters of the initial population. We have chosen the case that requires more precision ($E_{\text{req}} = 0.01$). The comparison results between the two proposed algorithms for the first three benchmarks are shown in Figs. 7, 8 and 9.

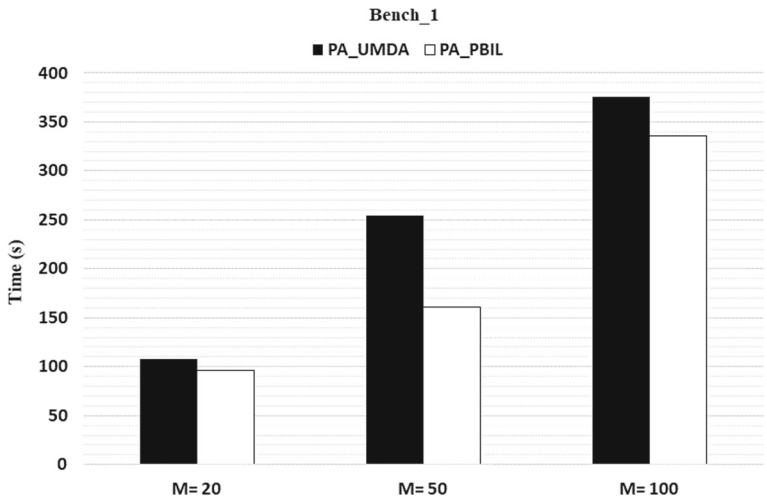

Fig. 7 Comparison between the processing times of the two algorithms (bench_1)

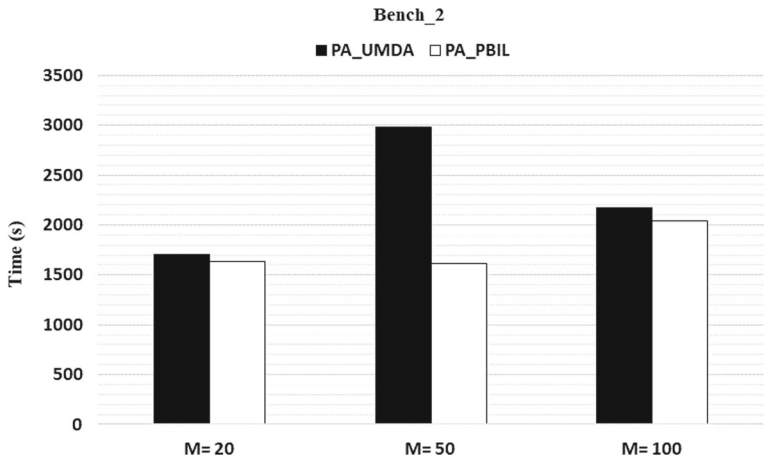

Fig. 8 Comparison between the processing times of the two algorithms (bench_2)

We varied the size of the initial population ($M = \{20, 50, 100\}$) to study the behavior of the two algorithms, we noted for the three benchmarks that for:

- $M = 20$: The processing times of the two algorithms are very close with a small advantage for the PA_PBIL algorithm;
- $M = 50$: A large gap between the processing times of the two algorithms, the PA_PBIL algorithm is more efficient;
- $M = 100$: The gap between the processing times of the two algorithms decreases for all three benchmarks.

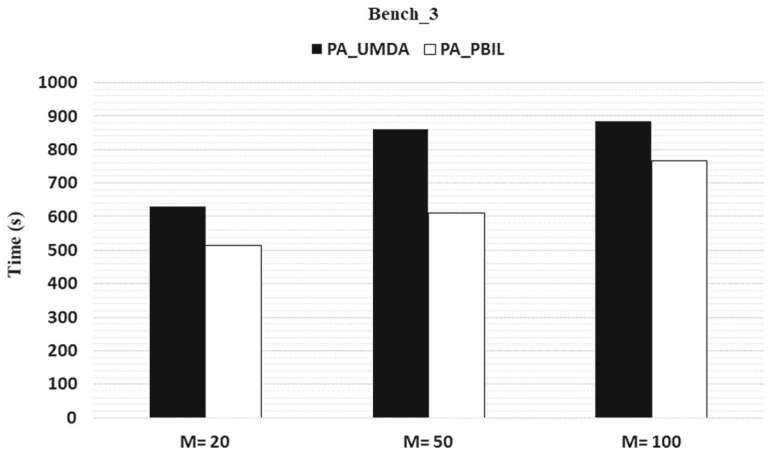

Fig. 9 Comparison between the processing times of the two algorithms (bench$_3$)

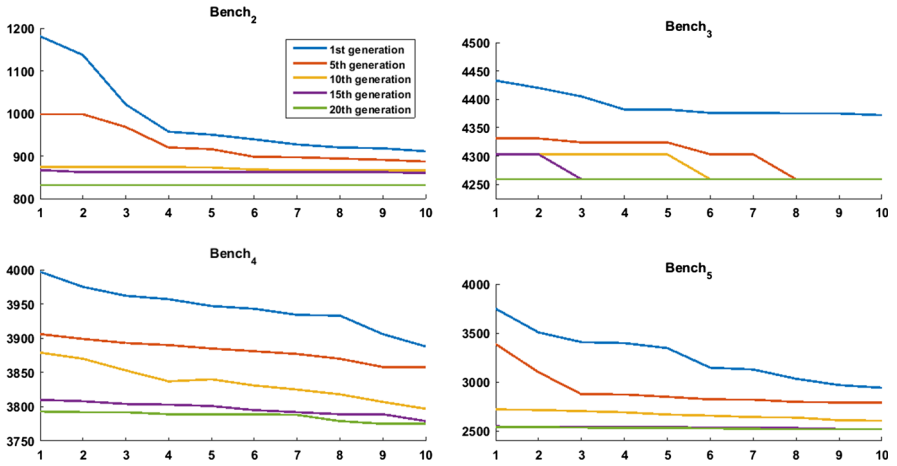

Fig. 10 Costs curve of the best selected solutions ($S = 10$) at different generations

6.1 Behavior and Stability of the Proposed EDA

To demonstrate the stability and convergence of the EDA algorithm for our problem, first, we plot the costs of the best-selected solutions in decreasing order ($S = 10$) at different generations (we have selected the generations: 1, 5, 10, 15, 20.) (Fig. 10). Second, we also represent the E_{req} partitions between the blocks of all individuals at different generations for the Bench$_2$ (Fig. 11). Individuals are ranked in decreasing order of their costs. The simulation results displayed in Figs. 10 and 11 clearly show that from generation to generation the algorithm tends toward the best solution, and the search space gets smaller and smaller. For example, in Fig. 10, for bench$_2$ and

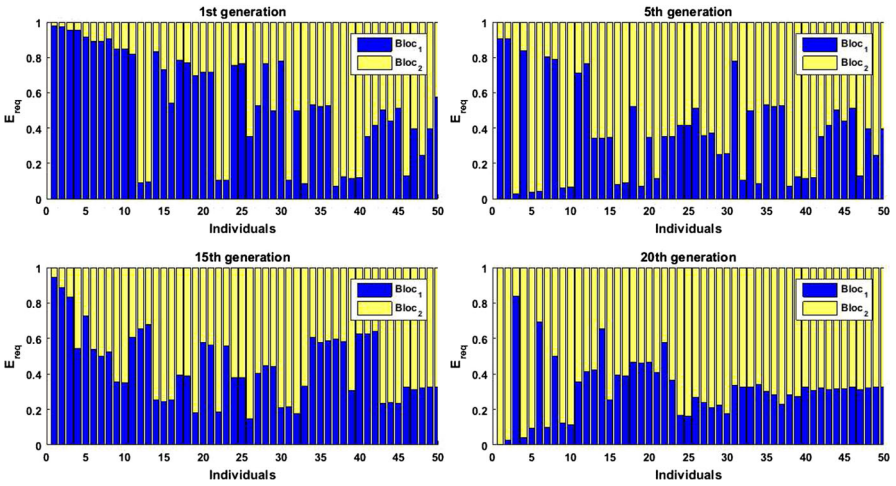

Fig. 11 E_{req} partitions between the blocks at different generations (Bench$_2$)

bench$_4$, we see that the cost curve of the generation k is strictly less than the one of the generation $k-5$. That said, if we set $C_i(k)$ as the cost of solution i at the generation k, we can see:

$$C_i(k) < C_i(k-5)\ \forall i = 1, \ldots, 10, \forall k = 5, \ldots, 20, \tag{20}$$

whereas for bench$_3$ the algorithm reaches the best solution at the $5th$ generation, so:

$$C_i(k) \leq C_i(k-5)\ \forall i = 1, \ldots, 10, \forall k = 10, \ldots, 20. \tag{21}$$

For bench$_2$, the simulation results show that the most promising region is located in the partition ($block_1 = [0.2, 0.4]$, $block_2 = [0.6, 0.8]$). In Fig. 11, we can see that in the first generation (random distribution) only 5 individuals are located in this region. Then, as the generations progress, the algorithm tends toward the most promising region. In the 5th generation, the number of individuals in this region increases to 14, then to 22 in the 15th generation, finally to 29 in the 20th generation.

As shown in Figs. 10 and 11, by sampling from a probability model built on the promising solutions in the parent generation, EDA can generate a new population in the promising region for the next generation. Hence, it avoids unnecessary repetitive steps and provides more search speed by making the search space smaller and smaller. This is the main reason for using EDA in this paper.

For this analysis, we used the PA_UMDA algorithm with the following inputs:

- Required error :
  - bench$_2$ and Bench$_4$: $E_{\text{req}} = 0.1$;
  - Bench$_3$: $E_{\text{req}} = 0.01$;
  - Bench$_5$: $E_{\text{req}} = 0.9$;

– Input signal: $x \in [-100, 100]$;
– Number of individuals in the initial population: $M = 50$;
– The best individuals selected at each iteration : $S = 10$.

6.2 Impact of the Proposed Approach on the Filter's Frequency Response

To show that our approach does not affect the filter characteristics, we plotted the frequency response of the bench$_3$ before and after the application of our method, for this comparison we used the precision analysis method presented in [20] with $E_{\text{req}} = 0.1$, the two curves presented in Fig. 12 are satisfactorily identical, and the difference is negligible (less than $|10^{-4}|$).

6.3 Practical Example Showing the Impact of Our Proposed BWO Approach Upon the Filter Characteristics

To demonstrate the efficiency of our method in a specific case study, we have established a simulation analysis of a practical filter used in DPLL (Digital Phase Lock Loop) applications [43]. DPLL is important device components in Communications Networks. They can be used to demodulate a signal, recover a signal from a noisy communication channel, generate a stable frequency at multiples of an input frequency (frequency synthesis), or distribute precisely timed clock pulses in digital logic circuits such as microprocessors.

Our approach consists of applying the BWO method individually for each block in direct form of an IIR parallel filter; therefore, to prove the efficiency of the adapted method, it is sufficient to apply the latter on a direct form.

To show that our BWO method keeps the filter characteristics and does not affect the filtering quality, we filter the same noisy signal with the IIR Butterworth filter

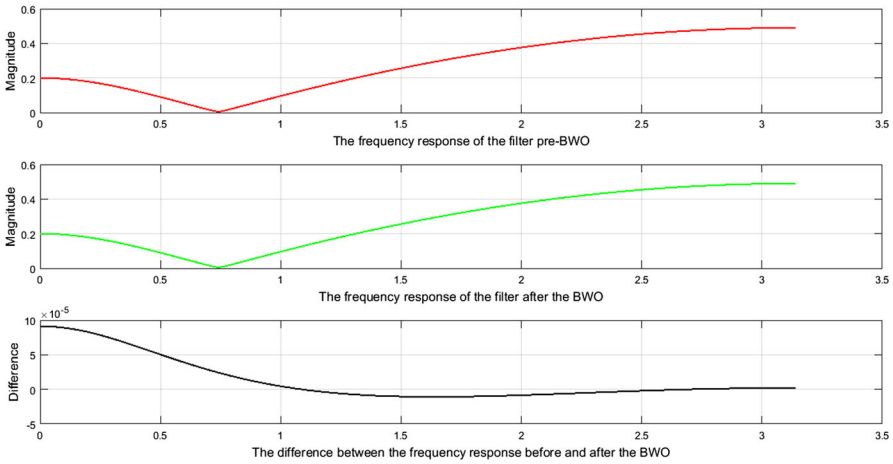

Fig. 12 Frequency response of the filter before and after the application of our approach (bench$_3$)

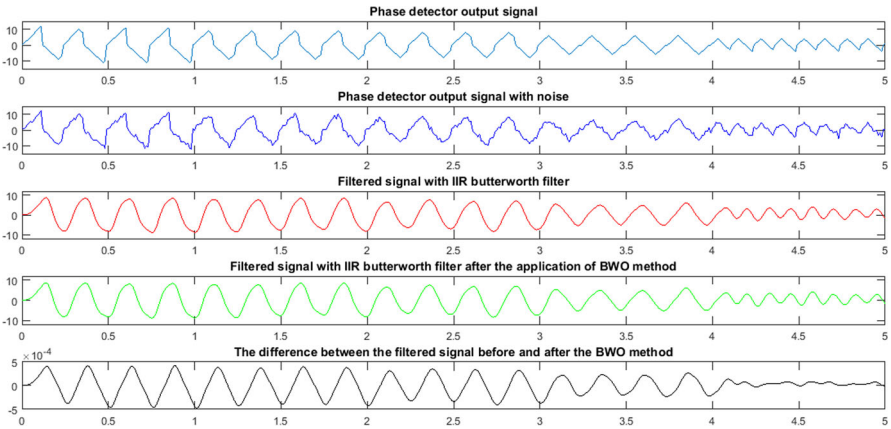

Fig. 13 Output signal filter before and after the application of our approach (Cut-off frequency 3 Hz)

(implemented on MATLAB according to the standard [43]) on both cases; before and after applying our BWO method. We compare the two filtered signals as well as the positions of the poles and zeros in both cases. In this simulation, we study the filtering of a signal coming from a phase detector with an input frequency of 8 KHz; we add a noise to this signal with a signal-to-noise ratio (SNR) of 1 dB. To test our BWO approach in several scenarios, we have varied the filter cut-off frequency (3, 6, 12 Hz). The simulation results show that our approach does not affect the filter characteristics. For the three different cases, we display: the phase detector output signal, noisy phase detector output signal, the filtered signal before and after the application of the BWO method, and the difference between the two filtered signals; we also display the position of the poles and zeros before and after the application of our BWO method. The results for each case are discussed as follows :

– For the cutoff frequency (3 Hz): Fig. 13 shows that the two signals are sufficiently identical; the difference is negligible (less than $|5 \times 10^{-4}|$). Figure 14 shows the position of the poles and zeros in both cases; we notice that the filter keeps its stability and preserves the position of the poles and zeros;

– For the cutoff frequency (6 Hz): the two filtered signals before and after using the BWO method are even closer in this case and the difference is less than $|2 \times 10^{-4}|$ (Fig. 15). The filter keeps the poles positions in the unit circle, hence preserving the filter stability, with a slight shift in the positions of the zeros (Fig. 16);

– For the cutoff frequency (12 Hz): we notice a slight difference that does not overtake $|10^{-4}|$ between the two filtered signals before and after using the BWO method (Fig. 17). We also notice just a tiny shift in the zeros position, but the filter remains stable (Fig. 18).

The precision analysis of the BWO method allows representing the fractional part of the filter coefficients with the minimum number of bits while respecting the constraints of precision and stability of the filter. Table 4 displays the filter coefficients and their

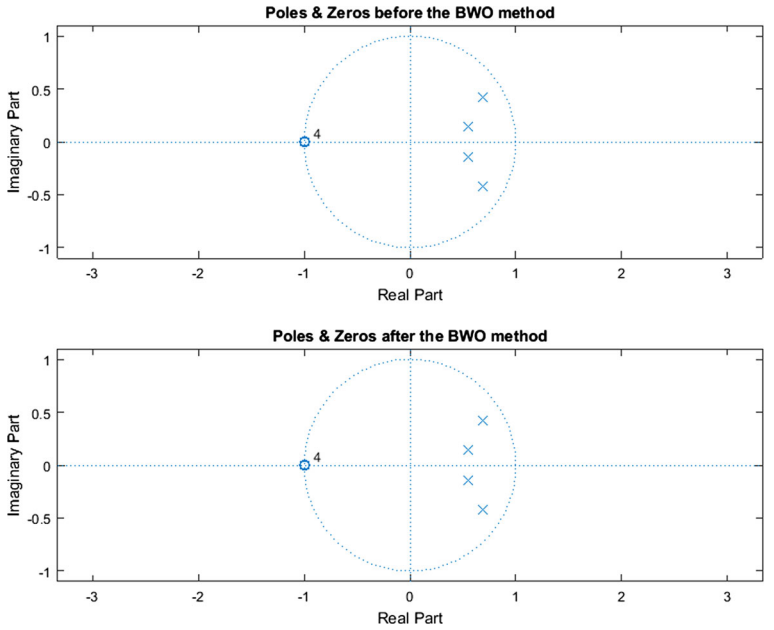

Fig. 14 Poles & Zeros positions before and after the application of our approach (cutoff frequency 3 Hz)

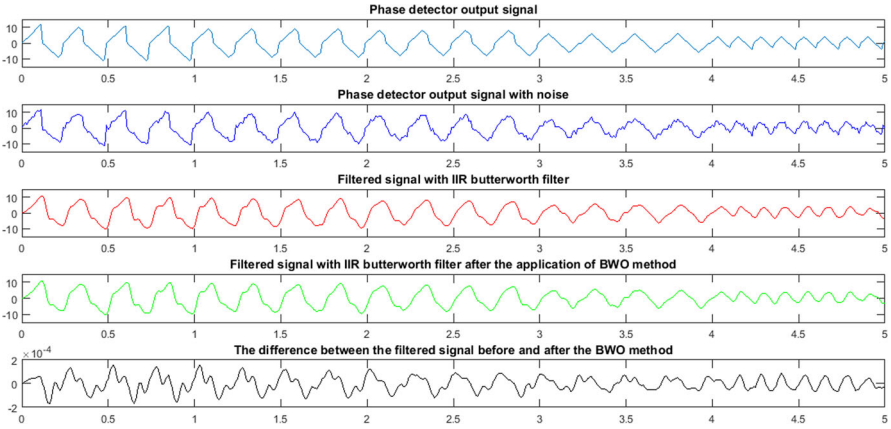

Fig. 15 Output signal filter before and after the application of our approach (Cut-off frequency 6 Hz)

bit-widths calculated through our approach as well as the quantified coefficients. The results are obtained with the following inputs:

- Required error : $E_{\text{req}} = 0.01$;
- Cut-off frequency: 3 Hz;
- Input signal: Noisy phase detector output (with SNR $=$ 1 dB), considering the input frequency at 8 KHz.

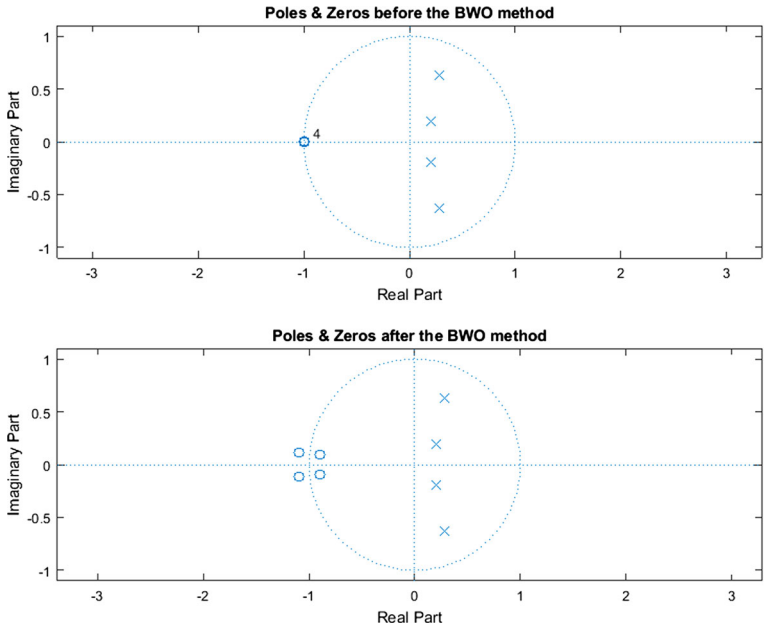

Fig. 16 Poles & Zeros positions before and after the application of our approach (cutoff frequency 6 Hz)

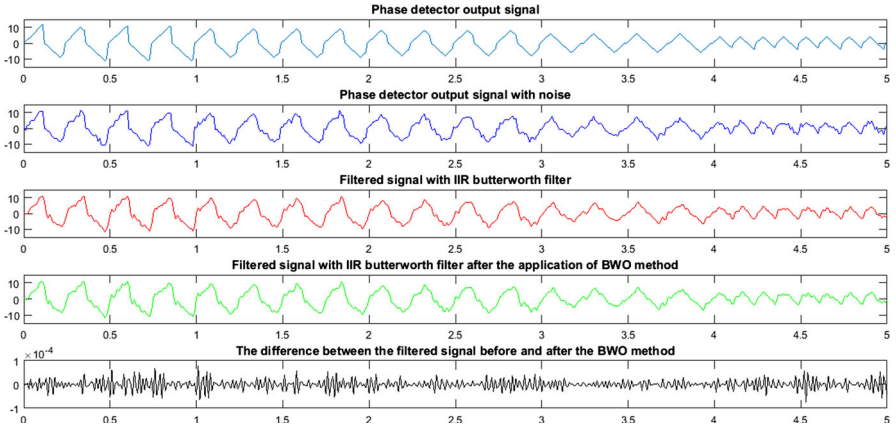

Fig. 17 Output signal filter before and after the application of our approach (Cut-off frequency 12 Hz)

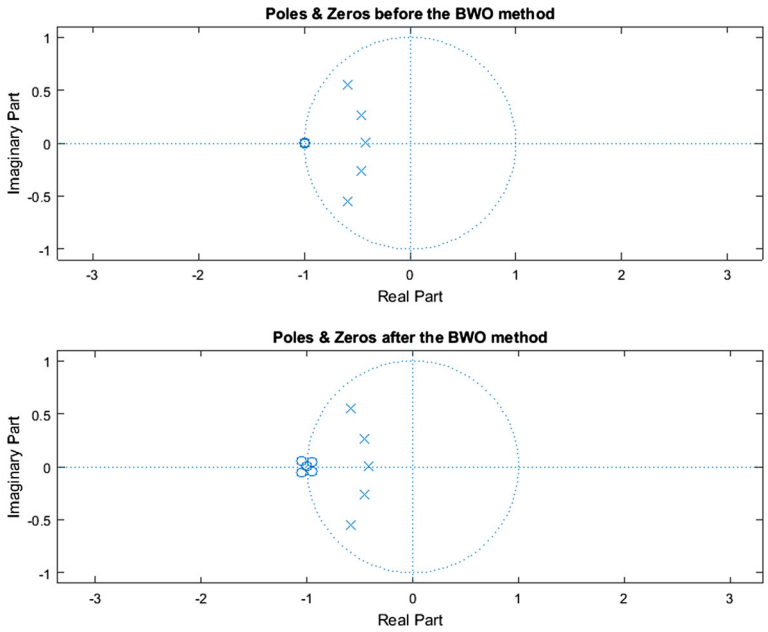

Fig. 18 Poles & Zeros positions before and after the application of our approach (cutoff frequency 12 Hz)

Table 4 Quantified coefficients after the application of our approach

Coefficient index	Filter coefficients	FBs allocation	Quantified coefficients
b_i	0.00394042601163910	18	0.00394058227539063
	0.0157617040465564	16	0.0157623291015625
	0.0236425560698346	17	0.0236434936523438
	0.0157617040465564	16	0.0157623291015625
	0, 00394042601163910	18	0.00394058227539063
a_j	1	0	1
	−2.46223993650885	18	−2.46223831176758
	2.46461621337966	16	2.46461486816406
	−1.14638710120606	18	−1.14638900756836
	0.207057640521471	18	0.207057952880859

7 Conclusion

This paper proposes a new precision analysis method designed for parallel IIR filter. In contrast with the existing literature, we suggest sharing the required error using a metaheuristic known as the Estimation of Distribution Algorithm. We have shown that when the required error is distributed through the blocks, our method outperforms existing approaches thanks to its flexibility that uses all required and distributed errors

and to the introduction of an heterogeneity between blocks in terms of the error's tolerance capacity. Actually, this newly adds to the heuristic optimization efforts dedicated in particular to the architectural aspect. Indeed, our results show that the application of the EDA approach, with appropriate precision analysis methods, brings improvements reaching the cost of up to 37%. Furthermore, our proposal outperforms existing method regardless of the adopted forward precision analysis approach. Within the EDA-based solution, we made a comparison between the convergence speed of the PA_PBIL algorithm compared to the PA_UMDA one. To demonstrate the efficiency of our method in a specific case study, we have established a simulation analysis of a practical filter used in DPLL applications [43]. This analysis consists of studying the filter output upstream and downstream to highlight that our method doesn't affect the filter characteristics and stability.

Funding The authors did not receive support from any organization for the submitted work.

Data Availability The data that support the findings of this study are the benchmarks provided in [36] which have been generated by the authors.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. A. Aggarwal, T.K. Rawat, D.K. Upadhyay, Optimal design of l 1-norm based iir digital differentiators and integrators using the bat algorithm. IET Signal Process. **11**(1), 26–35 (2016)
2. N. Agrawal, A. Kumar, V. Bajaj, A new method for designing of stable digital iir filter using hybrid method. Circuits Syst. Signal Process. **38**(5), 2187–2226 (2019)
3. C.R. Babu, B.H. Kumar, S.L. Kumari Narava, A heuristic approach of designing parallel filter using an enriched pole placement technique. Order **6**(4) (2018)
4. S. Baluja, Population-based incremental learning. A method for integrating genetic search based function optimization and competitive learning. Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Dept Of Computer Science (1994)
5. S. Baluja, R. Caruana, Removing the genetics from the standard genetic algorithm, in *Machine Learning Proceedings 1995*. (Elsevier, 1995), pp. 38–46
6. S. Baluja, S. Davies, *Combining multiple optimization runs with optimal dependency trees* (Carnegie-mellon univ pittsburgh pa dept of computer science, Tech. rep., 1997)
7. R. Bellal, E.S. Lamini, H. Belbachir, S. Tagzout, A. Belouchrani, Improved affine arithmetic-based precision analysis for polynomial function evaluation. IEEE Trans. Comput. **68**(5), 702–712 (2018)
8. D. Boland, G.A. Constantinides, Bounding variable values and round-off effects using handelman representations. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **30**(11), 1691–1704 (2011)
9. G. Caffarena, C. Carreras, J.A. López, Á. Fernández, Sqnr estimation of fixed-point dsp algorithms. EURASIP J. Adv. Signal Process. **2010**(1), 171027 (2010)
10. A. Carini, V.J. Mathews, G.L. Sicuranza, Sufficient stability bounds for slowly varying direct-form recursive linear filters and their applications in adaptive iir filters. IEEE Trans. Signal Process. **47**(9), 2561–2567 (1999)
11. J.L.D. Comba, J. Stol, A ne arithmetic and its applications to computer graphics, in *Proceedings of VI SIBGRAPI (Brazilian Symposium on Computer Graphics and Image Processing)* (Citeseer, 1993), pp. 9–18

12. G.A. Constantinides, P.Y. Cheung, W. Luk, The multiple wordlength paradigm, in *The 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01)* (IEEE, 2001), pp. 51–60
13. G.A. Constantinides, G.J. Woeginger, The complexity of multiple wordlength assignment. Appl. Math. Lett. **15**(2), 137–140 (2002)
14. J. De Bonet, C. Isbell, P. Viola, Mimic: finding optima by estimating probability densities. Adv. Neural Inf. Process. Syst. **9**, 424–430 (1996)
15. A. Fernandez-Vazquez, G.J. Dolecek, Generalized chebyshev filters for the design of iir filters and filter banks. Circuits Syst. Signal Process. **33**(7), 2237–2250 (2014)
16. G. Harik, et al., Linkage learning via probabilistic modeling in the ecga. IlliGAL report **99010** (1999)
17. K. Horváth, B. Bank, Optimizing the numerical noise of parallel second-order filters in fixed-point arithmetic. J. Audio Eng. Soc. **67**(10), 763–771 (2019)
18. K.I. Kum, W. Sung, Combined word-length optimization and high-level synthesis of digital signal processing systems. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **20**(8), 921–930 (2001)
19. Lamini, E.s., Bellal, R., Tagzout, S., Belbachir, H., Belouchrani, A.: Enhanced bit-width optimization for linear circuits with feedbacks. In: 2014 9th International Design and Test Symposium (IDT), pp. 168–173. IEEE (2014)
20. E.S. Lamini, S. Tagzout, H. Belbachir, A. Belouchrani, Precision analysis with analytical bit-width optimisation process for linear circuits with feedbacks. IET Circuits Dev. Syst. **12**(5), 563–570 (2018)
21. P. Larrañaga, J.A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*, vol. 2 (Springer, 2001)
22. D.U. Lee, A.A. Gaffar, R.C. Cheung, O. Mencer, W. Luk, G.A. Constantinides, Accuracy-guaranteed bit-width optimization. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **25**(10), 1990–2000 (2006)
23. K. Long, H. Chen, X. Li, Analysis and optimization for hardware implementation of sine/cosine with faithful rounding and monotonicity through piecewise quadratic polynomial. IEICE Electron. Exp. **18**, 20210158 (2021)
24. B. Lopez, Implémentation optimale de filtres linéaires en arithmétique virgule fixe. Ph.D. thesis (2014)
25. V. Magron, Interval enclosures of upper bounds of roundoff errors using semidefinite programming. ACM Trans. Math. Softw. (TOMS) **44**(4), 1–18 (2018)
26. V. Magron, G. Constantinides, A. Donaldson, Certified roundoff error bounds using semidefinite programming. ACM Trans. Math. Softw. (TOMS) **43**(4), 1–31 (2017)
27. D. Menard, G. Caffarena, J.A. Lopez, D. Novo, O. Sentieys, Fixed-point refinement of digital signal processing systems (2019)
28. H. Mühlenbein, The equation for response to selection and its use for prediction. Evol. Comput. **5**(3), 303–346 (1997)
29. H. Mühlenbein, G. Paass, From recombination of genes to the estimation of distributions i. binary parameters, in *International Conference on Parallel Problem Solving from Nature* (Springer, 1996), pp. 178–187
30. A.V. Oppenheim, J.R. Buck, R.W. Schafer, *Discrete-Time Signal Processing*, vol. 2 (Prentice Hall, Upper Saddle River, 2001)
31. G. Ott, E.A. Costa, S.J. Almeida, M.B. Fonseca, Iir filter architectures with truncation error feedback for ecg signal processing. Circuits Syst. Signal Process. **38**(1), 329–355 (2019)
32. Y. Pang, K. Radecka, Z. Zilic, An efficient hybrid engine to perform range analysis and allocate integer bit-widths for arithmetic circuits, in *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)* (IEEE, 2011), pp. 455–460
33. M. Pelikan, D.E. Goldberg, E. Cantu-Paz, Linkage problem, distribution estimation, and bayesian networks. Evol. Comput. **8**(3), 311–340 (2000)
34. J. Potsangbam, M. Kumar, Design and implementation of combined pipelining and parallel processing architecture for fir and iir filters using vhdl. Int. J. VLSI Des. Commun. Syst. **10**(4) (2019)
35. N. Revol, Introduction à l'arithmétique par intervalles (2001)
36. O. Sarbishei, Y. Pang, K. Radecka, Analysis of range and precision for fixed-point linear arithmetic circuits with feedbacks, in *2010 IEEE International High Level Design Validation and Test Workshop (HLDVT)* (IEEE, 2010), pp. 25–32
37. O. Sarbishei, K. Radecka, Z. Zilic, Analytical optimization of bit-widths in fixed-point lti systems. IEEE Trans. Comput. -Aided Des. Integr. Circuits Syst. **31**(3), 343–355 (2012)

38. C. Shi, R.W. Brodersen, Automated fixed-point data-type optimization tool for signal processing and communication systems, in *Proceedings of the 41st Annual Design Automation Conference* (2004), pp. 478–483
39. J.J. Shynk, Adaptive iir filtering using parallel-form realizations. IEEE Trans. Acoust. Speech Signal Process. **37**(4), 519–533 (1989)
40. D. Sidhu, J. Dhillon, Design of digital iir filter with conflicting objectives using hybrid predator-prey optimization. Circuits Syst. Signal Process. **37**(5), 2117–2141 (2018)
41. R. Singh, S.K. Arya, Genetic algorithm for the design of optimal iir digital filters (2012)
42. K.S. Tang, K.F. Man, S. Kwong, Z.F. Liu, Design and optimization of iir filter structure using hierarchical genetic algorithms. IEEE Trans. Ind. Electron. **45**(3), 481–487 (1998)
43. T. Technologies, Clocks for the synchronized network: Common generic criteria. GR-1244-CORE. **Issue 3. 2005** (2005). http://www.telcordia.com
44. H.L.N. Thi, S. Van Gerven, C. Jutten, D. Van Compernolle, Stability study for source separation in convolutive mixtures of two sources. Signal Process. **62**(2), 163–171 (1997)
45. J.T. Tsai, J.H. Chou, T.K. Liu, Optimal design of digital iir filters by using hybrid taguchi genetic algorithm. IEEE Trans. Ind. Electron. **53**(3), 867–879 (2006)
46. S. Vakili, J.P. Langlois, G. Bois, Enhanced precision analysis for accuracy-aware bit-width optimization using affine arithmetic. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **32**(12), 1853–1865 (2013)
47. Y. Yuan, C. Chen, X. Hu, S. Peng, Unlabeled data driven channel-wise bit-width allocation and quantization refinement. Aust. J. Intell. Inf. Process. Syst. **16**(4), 9–16 (2019)