On the Fixed-Point Accuracy Analysis and Optimization of Polynomial Specifications

Omid Sarbishei and Katarzyna Radecka, *Member, IEEE*

***Abstract*—Fixed-point accuracy analysis and optimization of polynomial data-flow graphs with respect to a reference model is a challenging task in many digital signal processing applications. Range and precision analysis are two important steps of this process to assign suitable integer and fractional bit-widths to the fixed-point variables and constant coefficients in a design such that no overflow occurs and a given error bound on maximum mismatch (MM) or mean-square-error (MSE) and signal-to-quantization-noise ratio (SQNR) is satisfied. This paper explores efficient optimization algorithms based on robust analyses of MM and MSE/SQNR for fixed-point polynomial data-flow graphs. Experimental results illustrate the robustness of our analyses and the efficiency of the optimization algorithms compared to previous work.**

***Index Terms*—Fixed-point designs, polynomial specifications, precision analysis, range analysis, word-length optimization.**

I. Introduction

MANY DIGITAL signal processing (DSP) systems require an implementation of nonlinear datapaths on application-specific integrated circuit (ASIC)/field-programmable gate array (FPGA) hardware. Typically, a reference model of such systems is defined in a high-resolution floating-point format, e.g., 64-b double precision. Therefore, it is feasible to assume that the reference model has infinite precision. Since the direct implementation of large floating-point systems on ASIC or FPGAs is not affordable in terms of hardware cost, designers often have to convert the initial design into an efficient fixed-point representation such that a set of given accuracy constraints, e.g., maximum spot error and (or) signal-to-noise ratio, are satisfied (verification). Furthermore, the hardware cost should be minimized (optimization).

Arithmetic functions in nonlinear datapaths are mostly represented by polynomials, which are suitable for both optimization and verification purposes [1]–[6]. In particular, some specific DSP applications require modular polynomial computations [1] performed over Z_2n. To aid the design process, the modular Horner expansion diagram [3], [54] can be used to canonically represent a polynomial over Z_2n, i.e., scalars modulo 2^n, and it can be utilized for optimization and verification purposes. The main disadvantage is that this kind of approach cannot handle imprecise datapaths like fixed-point representation, which has vast applications in DSP domains [7].

Fixed-point polynomial specifications, which are more pronounced in DSP applications compared to modular arithmetic circuits, can be optimized or verified at high levels of abstraction in two consecutive stages, i.e., algorithmic and imprecision (Fig. 1). At the algorithmic level, approaches such as scheduling [4] and algebraic factorization [4], [5] are utilized to generate a data-flow graph (DFG) of multipliers or adders, while maintaining a low hardware cost. At the imprecision level, analyses of range and precision are applied to set the integer and fractional bit-widths (IB and FB) of fixed-point variables and constant coefficients such that the results exhibit no overflow. Furthermore, a given error bound on maximum mismatch (MM) or mean-square-error (MSE), and, in consequence, signal-to-quantization noise ratio (SQNR) must be satisfied (Fig. 1). After the word-length-optimization (WLO) algorithms are applied, the circuit goes through lower level logic synthesis techniques such as those described in [43]–[45] and [52].

Note that the error (mismatch) y_e of the reference polynomial y is defined as the difference between y and its fixed-point implementation y_{fixed}

$$y_e = y - y_{\text{fixed}}. \tag{1}$$

The error metrics MM and MSE are defined as

$$MM = \max(|y_e|) \;\; MSE = \mathrm{E}\,(y_e^2) \tag{2}$$

where $\mathrm{E}(x)$ and $|x|$ return the expected and absolute values of x, respectively. The values of MM and MSE in (2) are evaluated over all the possible combinations of inputs. The error measure MSE, which captures the notion of power-of-noise, is usually of higher importance in DSP applications compared to the error measure MM, which indicates the largest possible spot error [8], [9]. Note that MSE can be used to obtain SQNR, which represents the classical signal-to-noise concept. It is, in particular, required that the MM and MSE in (2) satisfy a specific maximum bound as follows:

$$MM < MM_{\text{bound}} \tag{3}$$

$$MSE < MSE_{\text{bound}} \tag{4}$$

where $MM_{\text{bound}}/MSE_{\text{bound}}$ is the maximum bound on MM/MSE. The use of MM in the analysis of linear and

Manuscript received March 21, 2012; revised June 26, 2012 and October 22, 2012; accepted December 17, 2012. Date of current version May 15, 2013. This paper was recommended by Associate Editor L. Pozzi.

The authors are with the Department of Electrical and Computer Engineering, McGill University, Montreal, QC H3A 2K6, Canada (e-mail: omid.sarbishei@mail.mcgill.ca; katarzyna.radecka@mcgill.ca).

Digital Object Identifier 10.1109/TCAD.2013.2238290

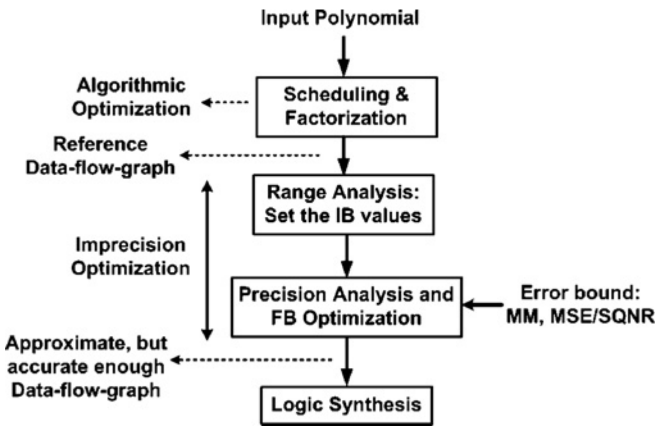
Input Polynomial
Algorithmic Optimization
Scheduling & Factorization
Reference Data-flow-graph
Range Analysis: Set the IB values
Imprecision Optimization
Precision Analysis and FB Optimization
Error bound: MM, MSE/SQNR
Approximate, but accurate enough Data-flow-graph
Logic Synthesis

Fig. 1. Optimization flow for imprecise polynomial specifications.

nonlinear circuits was addressed in [11], [12] and [13], [14], [50], respectively, while the methods in [6] and [16] target the analysis of MSE for linear and nonlinear circuits.

The previous work on optimizing the FB values of fixed-point variables in a nonlinear DFG, while satisfying the error bound, (3) and (4), is mostly based on the heuristic iterative methods used to search among different word-length assignments to the variables [10], [16], [37]. Such solutions often do not provide simultaneously the flexibility in controlling the bit-widths of coefficients and variables, which limits the efficiency of the optimization algorithm [48]. In this paper, we make use of arithmetic transform (AT) [25] as a unified analytical framework to compute the error measures MM and MSE for polynomial DFGs.

The proposed analysis of MM always overestimates the exact absolute value of an error, making it safe and robust (pessimist approach). Furthermore, we propose techniques to widely reduce the amount of error overestimation such that more realistic and accurate computations of MM could be achieved. This paper is the first to use AT for the MM analysis of nonlinear recursive DFGs. Furthermore, the use of AT for the MSE analysis of polynomial DFGs with possible feedbacks is studied for the first time. In fact, AT makes it possible to parameterize the error of the DFG with respect to the quantization error of intermediate variables along with constant coefficients without using Monte Carlo simulations.

Our error analyses lead to a novel formula as an optimization method for imprecise data-flow graphs (OMID). The formula is useful for the simple assignment of suitable FB values to the variables, while minimizing the hardware cost and satisfying the error bounds. It handles both error measures MM and MSE, and is very fast. In fact, it returns the deterministic optimal solution with the complexity of $O(n)$, where n is the total number of input and intermediate variables. The time complexity can be further improved to $O(1)$, if vector computations are available. The proposed WLO formula, OMID, is an extension to the truncation techniques presented in [14], which are only applicable to the error measure MM and cannot handle constant multipliers (adders) and recursive DFGs. Our WLO formula not only is implementable on the top of the error analysis based on AT (this paper), but also can be incorporated into the conventional error analyses that make use of linear approximations to represent the error of the DFG, e.g., AA [16]. This is another important contribution of this paper compared to the classical WLO solutions, which are mostly based on iterative heuristics. For instance, the approach in [61] uses an iterative method based on derivative computations to find the optimal point in the continuous domain, and finally, a ceiling function is used to round-up the results to the closest higher integers. This approach gives a near-optimal solution in the integer domain.

Our main contributions in this paper illustrate the use of AT as a unified analytical framework for the robust computation of MM and MSE for polynomial DFGs with possible feedbacks without using simulations. Furthermore, we introduce a novel convex integer programming scheme and its deterministic solution with a linear complexity to perform WLO for polynomial DFGs with respect to the error measures MM and MSE.

The remainder of this paper is organized as follows. Section II discusses the related work. Section III presents the use of AT as a unified analytical framework to compute MM and MSE. Section IV presents the proposed analytical WLO algorithm to set the FB values of variables and constant coefficients, while satisfying (3) and (4). Section V presents the experimental results. Section VI concludes this paper.

II. Related Work

Addressing the range, MM and MSE/SQNR analyses of polynomial DFGs can be done in several ways, which, in general, can be categorized as dynamic and static analyses. Dynamic analysis methods [17]–[21] including simulation are very slow and nonrobust. They often lead to under-allocation of IB and FB values, which confines their applicability. Static analysis approaches, which have recently gained much interest, are mostly faster than dynamic methods. They provide safe, but possibly pessimist results.

The static analysis of ranges of polynomial DFGs, which aims to set the IB of fixed-point variables in the DFG (Fig. 1), has been well studied in the literature [22]–[24]. Interval arithmetic (IA) and affine-arithmetic (AA) [22] are among the most straightforward solutions. AA mostly gives tighter ranges compared to IA, resulting in lower values of IB, and hence, lower hardware costs [56]. However, AA is slower than IA. Tighter ranges can be obtained using more sophisticated approaches, such as satisfiability modulo theory [23], AT [24], multielement IA [62], and polynomial chaos expansion [63], which usually come with higher computational costs compared to IA and AA alone. It has been shown in [23] that regarding range analysis of polynomial DFGs, which do not involve operators such as division, AA is mostly sufficient to find low values of IBs for the fixed-point variables, while avoiding the overflow issue. Hence, in this paper, we make use of the AA approach [22] for the range analysis of polynomials.

The MM computation of polynomial DFGs with respect to some given FB values for input, intermediate variables and coefficients can efficiently be achieved, among others, using AA

[22], [46], [47], [56] or AT paired with a branch-and-bound (BB) search [25], [41], [49]. The solutions in [13] and [22] have also utilized AA and AT, respectively, to propose heuristic search algorithms to set the bit-widths of input variables and constant coefficients such that a given bound on MM as in (3) is satisfied. The method in [13], which is based on AT, provides much more precise and accurate analyses of MM, compared to AA in [22] and [47]. Hence, a much lower hardware cost is achieved after WLO based on AT than that by AA [13]. The optimization in [14] presents more efficient techniques to analytically truncate the variables such that the condition in (3) is satisfied. However, this method can only handle DFGs with multipliers and adders with variable (parametric) inputs. Hence, constant multipliers and adders are not studied. The truncation techniques in [14] for parametric multipliers are applied to the DFGs with mixed coordinate rotation digital computer (CORDIC) units and to polynomial computations [50]. The analytical WLO technique presented in this paper, which provides a unified framework for both MM and MSE error measures, extends the analyses in [14] to cover all the possible arithmetic operations in a polynomial DFG, including constant and parametric multipliers and adders. Note that if the DFG involves CORDIC units in addition to multipliers and adders, the approach in [50] can be made use of, to first covert the whole DFG into a global polynomial using AT, and then the analyses in this paper can be used to handle the accuracy analysis and optimization of the DFG.

While analysis of MSE/SQNR has been investigated extensively for linear-time-invariant circuits with or without feedbacks, e.g., [6], [10], [15], [36], [40], [48], [55], further studies have extended the work to nonlinear circuits [16], [31], [32], [37], [53] based on the noise model in [33]. The approaches presented in [16], [31], [32], [37], and [53] are mostly based on perturbation methods and linear approximations of the original nonlinear system to propagate the error. The solutions in [16] and [37] have gone one step further. By making use of linear approximations, they have also proposed methods for covering nonlinear circuits with possible feedbacks. While the approach in [16] uses AA to linearly represent the error of the DFG with respect to the quantization errors, the solution in [37] makes use of a time-varying linear model to propagate noise and represent the MSE of the DFG. The parametrization methods in [16] and [37] require the quantized coefficients be given. Hence, the Monte Carlo simulation should be re-executed, if, for example, we change the FB of coefficients for the purpose of optimization. This is not the case with the parametrization based on AT, since it does not rely on Monte Carlo simulations and it can parameterize the error of the DFG with respect to the quantization error of variables along with constant coefficients. On the plus side, the MSE analyses based on Monte Carlo simulations, e.g., [16] and [37], are more generic and not limited to polynomial specifications like AT. It is notable that while AT is limited to polynomial specifications, it can handle datapaths with CORDIC units as well [50], and hence, it can cover a variety of nonlinear and nonpolynomial computations. Moreover, unlike the methods in [16] and [37], which make use of linear approximations, AT can take into account the higher order effects of signals and quantization errors.

III. MM and MSE Analyses

In this section, we provide an error analysis in terms of MM and MSE. The analysis is capable of handling both round-to-nearest and truncation types of quantization.

First, in Section III-A, we present an approach to compute MM by safely overestimating the absolute value of an error (mismatch). In particular, Lemma 1 addresses this problem. Furthermore, we propose a technique to lower the overestimations to achieve more accurate computations of MM. Lemma 2, Corollary 1, and Lemma 3 tackle that issue. In Section III-B, we extend the MM analysis to cover DFGs with possible feedbacks. Section III-C presents the analysis of MSE using AT, which does not rely on Monte Carlo simulations. Section III-D discusses the analysis of MSE for nonlinear recursive DFGs.

A. Error Representation and MM Computation

The reference polynomial y can be represented by AT [25] as follows:

$$y = \sum_{i=1}^{M} a_i \times m_i$$

where a_i is a real number, $m_i = (\prod_{j=1}^{L} A_j^{P_{i,j}})$ is the ith monomial of the polynomial y, $i = 1, \ldots, M$, A_j is the jth primary input of y, $j = 1, \ldots, L$, $P_{i,j}$ is the degree of A_j in m_i, and it is a nonnegative integer. Note that the degree of the polynomial y is equal to $\max_{i=1,\ldots,M}\left(\sum_{j=1}^{L} P_{i,j}\right)$. The mismatch of y can also be expressed as a polynomial using AT as follows:

$$y_e = y - y_{\text{fixed}} = \sum_{i=1}^{\hat{m}} \hat{a}_i \times \widehat{m}_i \tag{5}$$

where $\hat{a}_i$ is a real number and

$$\widehat{m}_i = \left(\prod_{j=1}^{L} A_j^{P1_{i,j}}\right)\left(\prod_{j=1}^{L} A_{e\_j}^{P2_{i,j}}\right)\left(\prod_{k=1}^{Q} e_{q\_k}^{P3_{i,k}}\right) \times \left(\prod_{l=1}^{T} e_{c\_l}^{P4_{i,l}}\right). \tag{6}$$

Note that $\widehat{m}_i$ is the ith monomial of $y - y_{\text{fixed}}$, $i = 1, \ldots, \widehat{M}$, $P1_{i,j}$ is the degree of the primary input variable A_j in monomial $\widehat{m}_i$, $j = 1, \ldots, L$, and $P2_{i,j}$ is the degree of the quantization error of A_j, i.e., A_{e_j}, in $\widehat{m}_i$, while $P3_{i,k}$ is the degree of the kth intermediate variable's quantization error, i.e., e_{q_k} in $\widehat{m}_i$, $k = 1, \ldots, Q$. In addition to intermediate variables, there exist some constant coefficients in the DFG that need also to be quantized. Assume that we have T coefficients. Hence, corresponding to each coefficient we have a quantization error, i.e., e_{c_l} in (6). Furthermore, $P4_{i,l}$ is the degree of the quantization error e_{c_l} in $\widehat{m}_i$. Note that throughout the rest of this paper, we also denote the value of FB_k as the FB of the kth intermediate variable corresponding to the quantization error e_{q_k}. Moreover, FB_{Aj} represents the FB of the primary input A_j.

The expression of y_e in (5) is a general case due to the following reason. If the two arbitrary functions f_1 and f_2 are represented as (5), there exists another function f_3 (f_4) in the form of (5), which can represent the multiplication (addition) operation between f_1 and f_2. In fact, if both f_1 and f_2 consist of M monomials, then the multiplication (addition) result will include at most $M^2(2M)$ monomials in the form of (6).

Hence, the error function given by (5) consists of $\widehat{M}$ monomials, L primary input variables with their corresponding quantization errors, Q quantization errors corresponding to the intermediate variables that have been quantized, T constant coefficients that realize the values of $\hat{a}_i$ in (5), and the corresponding T number of quantization errors for each coefficient.

Lemma 1 describes the method for computing an overestimation of the absolute value of error.

Lemma 1: An overestimation of the absolute value of an error in (5), i.e., $|y_e|$, can be obtained by considering the worst case of intermediate variable quantization errors in (6), i.e., $e_{q\_k} = \pm 2^{-FB_k-1}$ for round-to-nearest quantization, and $e_{q_k} = 2^{-FB_k-1} \pm 2^{-FB_k-1}$ for truncation, where FB_k is the FB of the kth intermediate variable.

Proof: For the purpose of error analysis, considering the worst case of the absolute value of round-to-nearest quantization errors means that no matter what input combinations are, the correlated intermediate quantization errors are all at either their most positive, i.e., 2^{-FB_k-1}, or most negative values, i.e., -2^{-FB_k-1}. This assumption provides a safe overestimation when evaluating the absolute value of error, i.e., $|y_e|$. Similarly, for truncation the maximum and minimum values of the quantization error are $2^{-FB_k} = 2^{-FB_k-1}+2^{-FB_k-1}$, and $0 = 2^{-FB_k-1} - 2^{-FB_k-1}$, respectively. Note that we assume the continuous model maximum (minimum) values for the quantization errors to overestimate them. The amount of overestimation is mostly negligible since the number of truncation bits is typically high in real designs (more than 8 b) [14], [48]. ■

Lemma 1 indicates that by substituting the probabilistic quantization errors of intermediate variables into (6) with their deterministic worst-case scenarios, i.e., $e_{q\_k} = \pm 2^{-FB_k-1}$ for round-to-nearest and $e_{q\_k} = 2^{-FB_k-1} \pm 2^{-FB_k-1}$ for truncation, we overestimate the value of $|y_e|$. Hence, if we compute the value of $MM = \max(|y_e|)$ using the assumption in Lemma 1, then MM is safely overestimated. Through the rest of this subsection, we only look at the round-to-nearest type of quantization with the worst-case error values of $e_{q\_k} = \pm 2^{-FB_k-1}$, but the discussion is also applicable to truncation.

Lemma 2: Based on (5) and (6), we can deduce that there exist at least $L + T$ monomials m_i defined by (5) such that $P3_{i,k} = 0$ for $k = 1, \ldots, Q$.

Proof: Assume that none of the intermediate variables has been quantized, i.e., $P3_{i,k} = 0$, $k = 1, \ldots, Q$ in (6). Furthermore, assume that $e_{c\_l} = 0$, $l = 1, \ldots, T$, and $A_{e\_j} = 0$, $j = 2, \ldots, L$, which means that no quantization has been performed for all the coefficients and the primary inputs A_j, where $j = 2, \ldots, L$. Hence, the only quantization error that exists in (6) is A_{e_1}. This individual source of quantization error contributes to some errors at the output y_e, causing $y_e \neq 0$. Since $y_e \neq 0$, when the only source of quantization error is A_{e_1}, we should look for at least one single-variable monomial m_i in (6), which is only a function of A_{e_1}. Hence, in (6), the following equality holds for that particular single-variable monomial m_i: $P3_{i,k} = P4_{i,l} = 0$, $l = 1 \ldots, T$, and $k = 1, \ldots, Q$, and $P2_{i,j} = 0$, $j = 2, \ldots, L$, and $P2_{i,1} > 0$. A similar argument can be presented for the rest of $L - 1$ input quantization errors and the T coefficient quantization errors. Therefore, we deduce that at least $L+T$ monomials exist in the representation of y_e in (5), which are single-variable, and for which $P3_{i,k} = 0$ for all the values of k, i.e., $k = 1, \ldots, Q$. ■

Corollary 1: There exist Q monomials, e.g., m_i, $i = 1, \ldots, Q$ defined by (6) such that $P2_{i,j} = P3_{i,k} = P4_{i,l} = 0$, where $l = 1, \ldots, T$, $j = 1, \ldots, L$ and $k \in \{1, \ldots, Q\}\backslash\{i\}$, and $P3_{i,i} > 0$.

The proof follows that of Lemma 2.

Lemma 3: The overestimation of $|y_e|$ introduced by Lemma 1 can widely be reduced when the absolute value of the quantization error of intermediate variables, i.e., $|e_{q\_k}| = 2^{-FB_k-1}$ in (6), is much smaller compared to the absolute value of the quantization error of primary inputs and constant coefficients, i.e., when we have $|e_{c\_l}| \gg 2^{-FB_k-1}$ and $|A_{e\_j}| \gg 2^{-FB_k-1}$.

Proof: The conditions $|e_{c\_l}| \gg 2^{-FB_k-1}$ and $|A_{e\_j}| \gg 2^{-FB_k-1}$ imply that those monomials m_i in (6), which satisfy $P3_{i,k} > 0$, are much smaller than those with $P3_{i,k} = 0$. Hence, under such conditions, the value of $|y_e|$ is dominated by the monomials with $P3_{i,k} = 0$ (the ones that are independent of e_{q_k}). Due to Lemma 2, at least $L + T$ monomials in (5) satisfy $P3_{i,k} = 0$, and hence, they become dominant compared to the rest of the monomials since $|e_{c\_l}| \gg 2^{-FB_k-1}$ and $|A_{e\_j}| \gg 2^{-FB_k-1}$. Therefore, the amount of overestimation, which originates from overestimating the quantization errors of intermediate variables, i.e., $e_{q\_k} = \pm 2^{-FB_k-1}$ (see Lemma 1), becomes negligible, when the conditions $|e_{c\_l}| \gg 2^{-FB_k-1}$ and $|A_{e\_j}| \gg 2^{-FB_k-1}$ are satisfied. ■

Lemma 3 illustrates a promising optimization strategy relying on assigning a higher quantization error to the coefficients and primary inputs compared to the intermediate variables such that the amount of overestimation generated by converting the probabilistic intermediate quantization errors into their corresponding deterministic worst cases (see Lemma 1) becomes negligible.

Using the overestimation of $|y_e|$ based on Lemma 1, we aim to compute MM. The BB search algorithm presented in [25] makes it possible to determine MM defined by (2) when the FB values of constant coefficients, primary inputs, and intermediate variables are all given. Note that for a given value of FB for a constant coefficient, its quantization error e_{c_l} is a fixed value. The approach in [25] assumes the worst case of $A_{e\_j} = \pm 2^{-FB_{Aj}-1}$ for the input round-to-nearest quantization errors, where FB_{Aj} is the given FB of A_j when computing MM. Furthermore, the worst case of the quantization error of intermediate variables, i.e., $e_{q\_k} = \pm 2^{-FB_k-1}$ for round-to-nearest, and $e_{q\_k} = 2^{-FB_k-1} \pm 2^{-FB_k-1}$ for truncation, are taken into account. The BB search in [25] returns the value of MM and the particular values of primary inputs, i.e., A_j in (5), which contribute to the value of $MM = \max(|y_e|)$. Furthermore, the sign of each quantization error, i.e., "+" or "-," which contributes to MM, is found by the search.

Example 1: Consider the uniform cubic B-spline basic function B_0 defined as

$$B_0 = \frac{-1}{6}A_1^3 + \frac{1}{2}A_1^2 - \frac{1}{2}A_1 + \frac{1}{6} = \frac{-1}{6}(A_1 - 1)^3 \quad (7)$$

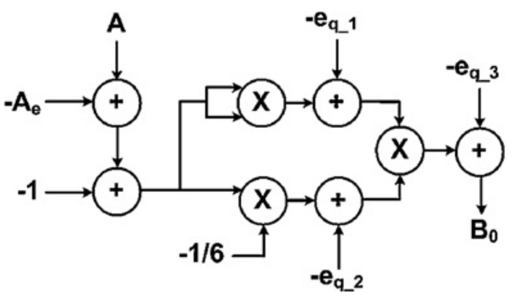

Fig. 2. Data-flow graph of Example 1.

where A_1 lies within the interval $[-1, 1]$. Assume that the FB of the input variable A_1 is fixed to 16 b using truncation. The DFG consists of an adder realizing $A_1 - 1$, two multipliers realizing $(A_1 - 1)^2$ and $\frac{-1}{6}(A_1 - 1)$, and a final multiplier to realize $\frac{-1}{6}(A_1 - 1)^3$, as depicted in Fig. 2. We assume that the FB of all the intermediate variables is set to 16 through truncation. Furthermore, the FB of the coefficient is set to 12 using round-to-nearest. By applying the BB search in [25], we find the overestimated value of $MM = 0.00075787$ at $A_1 = -1$, $A_{e\_1} = 2^{-16}$, $e_{q\_1} = 0$, $e_{q\_2} = 2^{-16}$, and $e_{q\_3} = 2^{-16}$. The exhaustive simulation returns the MM of 0.00073437. This means that our MM analysis provides 3.2% of overestimation.

Note that by making use of AA the MM is obtained as 0.0028, which corresponds to 281.28% of overestimation. □

B. MM Computation for Nonlinear Recursive DFGs

Most of the fixed-point DSP designs that involve feedbacks are LTI circuits, e.g., infinite impulse response (IIR) filters [12], lattice-structured filters [16], and delta-operator filters [35]. The analysis of MM for such circuits can be done as follows (using the proposed method discussed in Section III-A). First, we flatten the error of the DFG with respect to an initial number of loops, e.g., *#of loops=1*, using AT. Then, we compute MM using the BB search [25]. Next, we gradually increase the number of loops, until the computed value of MM converges. The number of loops required for convergence is dependent upon the position of poles in the LTI circuits. Note that the BB search in [25] involves a preprocessing step to exclude the linear or noncorrelated nonlinear variables from the search. Hence, it is scalable with respect to the size of the linear designs, even if it involves feedbacks.

For nonlinear DFGs with feedbacks, the problem is more complex since the convergence and stability conditions due to the infinite accumulation of error sources are not trivial to address. If the convergence is guaranteed, we can make use of the same strategy by gradually increasing the number of loops and flattening the error of the DFG using AT. However, for nonlinear DFGs with possible feedbacks, the BB search is not scalable with respect to the size of the design (more details are addressed in Section V). Hence, for the purpose of MM computation for recursive nonlinear DFGs, we make use of the following strategy.

1) Flatten the error of the DFG with respect to the given number of iterations (loops) using AT (5).
2) Use the triangle inequality to compute the maximum (minimum) value of y_e for the given AT representation as follows:

$$\max\left(y_e = \sum_{i=1}^{\widehat{M}} \hat{a}_i \times \widehat{m}_i\right) \leq \sum_{i=1}^{\widehat{M}} \max(\hat{a}_i \times \widehat{m}_i)$$

where each max() function in the above equation is computed using IA. By applying this method, the computation of MM can be done in polynomial time with respect to the size of the design and number of iterations since building the AT of the DFG becomes the bottleneck of the process. However, MM is possibly determined with higher overestimations compared to the method based on the BB search [25] due to the use of IA. In the experimental results presented in Section V, we show that AT paired with IA is more efficient than AA for computing MM. For instance, regarding Example 1, if we make use of AT paired with IA we obtain $MM = 0.000763$, which corresponds to 3.9% of overestimation.

C. MSE Analysis

The framework of error analysis in terms of MSE is analogous to the analysis of MM discussed in Section III-A. In fact, for the purpose of MSE analysis, we consider two scenarios.

1) *Scenario A:* The conditions in Lemma 3 are satisfied.
2) *Scenario B:* The conditions in Lemma 3 are not satisfied. For instance, the coefficient quantization errors are given as very small values.

In Scenario A, we make use of the overestimation described by Lemma 1, i.e., $e_{q\_k} = \pm 2^{-FB_k - 1}$ (substituting those probabilistic errors into their corresponding deterministic worst-case scenarios). That way, the value of MSE is computed with a negligible overestimation.

In the case of Scenario B, we apply the well-known uniform distribution noise model for the quantization errors. Furthermore, we assume that the quantization errors are independent not only of each other but also of the primary input variables. That way, the value of MSE is approximated with a good accuracy. Note that due to the aforementioned discussion in Lemma 1, we assume the continuous model maximum (minimum) values for the quantization errors to overestimate them. The amount of overestimation is mostly negligible since the number of truncation bits is typically high in real designs (more than 8 b) [14], [48]. Using such a model, the various moments of the quantization errors are calculated as follows:

$$E(e_q^k) = \frac{\sum_{j=0}^{k} (\min(e_q))^j \times (\max(e_q))^{k-j}}{k+1} \quad k \in \mathbb{N} \tag{8}$$

with the following values for $\min(e_q)$ and $\max(e_q)$ regarding truncation and round-to-nearest quantization:

Truncation: $\min(e_q) = 0$, $\max(e_q) = 2^{-FB}$
Rounding: $\min(e_q) = -2^{-FB-1}$, $\max(e_q) = 2^{-FB-1}$.

In order to compute MSE analytically, we also require the statistical characteristics of the primary inputs A_j in (5). If the primary input variables are uniformly distributed, then

(8) is also valid to find $E(A_j^k)$. There are also other simple cases of finding $E(A_j^k)$. For instance, when A_j has a Gaussian distribution with zero mean ($\mu = 0$) and the variance of σ^2, then we have

$$E(A_j^k) = \begin{cases} 0, & \text{if } k \text{ is odd} \\ \sigma^k (k-1)!!, & \text{otherwise} \end{cases}$$

where $n!!$ is the double factorial operator that is the product of every odd number from n to 1. We assume, in this paper, that the primary input variables are statistically independent of each other. However, even if that is not the case, given the statistical characteristics of the correlated primary inputs, i.e., $E(A_i^{k_1} A_j^{k_2})$ for all positive integer values of k_1 and k_2, we can compute MSE analytically using AT.

Given the statistical characteristics of the primary inputs and the quantization errors, it becomes easy to compute MSE using AT. In fact, based on the error function y_e represented by AT in (5), we can first build $AT(y_e^2)$, and then distribute the expectation function $E(\cdot)$ among all the monomials in $AT(y_e^2)$ to find MSE.

Example 2: Consider the polynomial $y = A^4$ with the DFG shown in Fig. 3. The reference input variable A has a Gaussian distribution with a zero mean and the variance of $\sigma^2 = \frac{1}{16}$ over the interval $[-1, 1]$. Hence, we have

$$\begin{aligned} &E(A^k) = \begin{cases} 0, & \text{if } k \text{ is odd} \\ \sigma^k (k-1)!!, & \text{otherwise} \end{cases} \\ &\Rightarrow E(A) = E(A^3) = E(A^5) = 0 \\ &\quad E(A^2) = 0.0625,\ E(A^4) = 0.0117,\ E(A^6) = 0.0037. \end{aligned}$$

Furthermore, all the variables are assumed to have 24 fractional bits after truncation. The quantization errors of both input and intermediate variables are also depicted in Fig. 3. Since the conditions in Lemma 3 are not satisfied for this example, we make use of the uniform distribution noise model for the quantization errors. The square error y_e^2 is computed as follows:

$$y_e^2 = (A^4 - ((A - A_e)^2 - e_{q\_1})^2 - e_{q\_2})^2.$$

If we build the AT of the square error in the above equation, we end up with 31 monomials. Next, based on the given statistical characteristics of the variables and quantization errors, and by distributing the expectation function $E(\cdot)$ among all the 31 monomials, we find $MSE = 1.5312 \times 10^{-15}$. Using exhaustive simulations, we also get $MSE = 1.4791 \times 10^{-15}$. Hence, the MSE analysis based on AT and the uniform distribution noise model results in around 3.5% overestimation of MSE. Note that by making use of the approaches in [16] and [37], we can also find an almost accurate value of MSE, e.g., $MSE = 1.5454 \times 10^{-15}$, which corresponds to around 4.6% of overestimation. The main difference between AT and the approaches in [16] and [37] is that unlike AT, those methods require a Monte Carlo simulation step in advance to parameterize the error of the DFG with respect to the quantization errors. Furthermore, they make use of linear approximations, while AT can arbitrarily take into account different moments and degrees of both signals and quantization errors. □

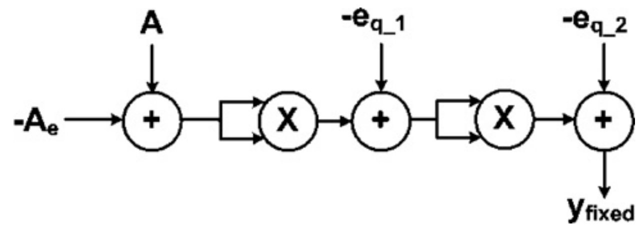

Fig. 3. Data-flow graph of Example 2.

D. MSE Computation for Nonlinear Recursive DFGs

The basic strategy to compute MSE for nonlinear recursive DFGs is analogous to the discussion presented in Section III-B. Namely, we flatten the error of the DFG with respect to a given number of loops using AT, compute MSE using the discussion in Section III-C, and then gradually increase the number of loops, until the computed MSE converges. Note that the determination of MSE does not rely on the BB search like the analysis of MM, and hence, is fast.

If the number of loops required for convergence is very high, it is possible to reduce the size of AT (#of monomials) by making use of linear approximations, which is mostly an accurate enough procedure [32]. Namely, during the process of flattening the DFG and building AT, we can ignore the monomials that generate higher order quantization errors. Using such an approach, more compact AT representations are obtained, but with the cost of potential negligible underestimations of MSE. For instance, in Example 2, the linearly approximated AT of the square error y_e^2 has only six monomials (instead of 31), and the same MSE of $MSE = 1.5312 \times 10^{-15}$ is obtained for the DFG. Furthermore, the experimental results in Section V show that regarding a nonlinear recursive eighth-order least-means-square (LMS) filter [64], the AT representation with and without linear approximations results in the same calculation of MSE.

IV. Analytical WLO Scheme

In this section, we present the proposed WLO algorithm with respect to the error measures MM and MSE. We map the problem into a convex integer programming problem, for which we propose a deterministic solution with linear complexity using a simple formula, OMID. The proposed WLO technique is not only implementable on the top of the error analyses based on AT (this paper), but also it can be merged with any other conventional MM and MSE analysis, which makes use of linear approximations with respect to the quantization errors to represent the error of the DFG.

First, in Section IV-A, we present the general linear scheme to approximate the error measures MM and MSE for the purpose of WLO. Regarding the error measure MM, the proposed WLO scheme can take into account both truncation and round-to-nearest types of quantization. However, for the error measure MSE, our WLO scheme is only capable of handling the round-to-nearest type of quantization since it corresponds to a zero mean. Section IV-B addresses the representation of hardware cost, and finally, Section IV-C presents our WLO formula, OMID.

A. Linear Approximations of MM and MSE

By considering the worst case of quantization errors e_{q_k} using truncation or round-to-nearest (Lemma 1) and ignoring their high order effects, i.e., $e^P_{q\_k}$, where $P \geq 2$, the error y_e becomes a linear function of e_{q_k}. Hence, the MM of the DFG is approximated as

$$MM \approx b_0 + \sum_{k=1}^{Q} b_k \times 2^{-FB_k}$$

where b_k, $k = 0, \ldots, Q$, are positive real coefficients that can be obtained using different methods such as AT paired with a BB search (Section III-A), AT paired with IA (Section III-B), or even AA [22]. If we make use of AT paired with the BB search, tighter computations of MM are achieved (more details are provided in the experimental results of Section V).

For the error measure MSE, if the quantization type is round-to-nearest, and the error of the DFG is linearly approximated with respect to the quantization errors, then we have

$$MSE = E(y_e^2) \approx \hat{b}_0 + \sum_{k=1}^{Q} \hat{b}_k \times 2^{-2FB_k}$$

where $\hat{b}_{0:Q}$ are all positive real coefficients that can be obtained using AT (this paper) or Monte Carlo simulations [16], [37]. Note that the linearly approximated MSE is a linear function of $E(e^2_{q\_k})$, which is proportional to 2^{-2FB_k}. Furthermore, the terms $2^{-FB_k}2^{-FB_j}$, where $k \neq j$, do not appear in the MSE representation since $E(e_{q\_k}) = E(e_{q\_j}) = 0$ for round-to-nearest.

Hence, we can represent both error measures MM and MSE in the following generic form:

$$Measure \approx f_0 + \sum_{k=1}^{Q} f_k \times 2^{-P_k FB_k} \tag{9}$$

where $f_{1:Q}$ is a positive real value, and $P_k = 1\,(P_k = 2)$ regarding the error measure MM (MSE).

Note that the linear approximation in (9) is mostly a valid assumption since the higher order effects of quantization errors are negligible values in practical cases. For instance, in our experiments, the approximated error measure as of (9) always matched the case, where the higher order effects of errors are also taken into account. This has been the case even for nonlinear recursive designs. In fact, the main source of inaccuracy when using a pure analytical model like AT to compute the error measures MM and MSE originates from the following two issues.

First, for the error measure MM, there is a complex correlation between the value of the primary inputs and the quantization error of the intermediate variables. We ignore this correlation by overestimating the absolute value of error (Lemma 1). Second, in the case of the error measure MSE, the uniform distribution assumption for quantization errors is not always valid.

B. Hardware Cost Representation

We represent the hardware cost as a linear function of the FB values, i.e., $\sum_{k=1}^{Q} a_k FB_k$, where a_k are positive real coefficients (weights). We can obtain the values of a_k in advance by either making use of manual approximations or by synthesizing a number of designs followed by a final linear curve-fitting step.

The hardware cost of $\sum_{k=1}^{Q} a_k FB_k$ is mostly acceptable to estimate the number of logic gates, delay, etc., in ASIC and FPGA designs. Note that if the target FPGA benefits from some dedicated embedded multipliers and adders with given bit-widths, then the WLO problem is translated into a resource allocation problem. Under such conditions, we first find the multipliers (adders) in the DFG, whose output quantization errors contribute the most to the error at the terminal output, i.e., the FB_k values that generate the highest values of f_k in (9). Then, we assign those multipliers (adders) to the available ones on the FPGA. Obviously, the FB_k with higher values of f_k must be assigned to the available multipliers (adders) with higher bit-widths. Finally, the remaining unassigned multipliers (adders) in the DFG should be assigned to logic gates, slices, look-up-tables, etc. Hence, for those arithmetic elements we perform a WLO using the hardware cost model $\sum_{k=1}^{Q} a_k FB_k$.

C. WLO Formula

Based on the error measure description (MM and MSE) in (9), and the hardware cost representation $\sum_{k=1}^{Q} a_k FB_k$, the novel WLO formula, OMID, is stated in Theorem 1.

Theorem 1: Assume the error measure (*Measure*) given by (9), where the positive values f_k, $k = 0, \ldots, Q$, the positive degrees P_k, and the maximum bound on the error measure MM (MSE), i.e., M_{bound}, are all given. The values of FB_k are all unknown, but assume that they are all set such that the following condition is satisfied:

$$A \times \frac{a_k}{2^{P_k} - 1} < f_k(2^{-FB_k})^{P_k} \leq A \times \frac{a_k}{1 - 2^{-P_k}} \qquad k = 1, \ldots, Q \tag{10}$$

where $A = (M_{\text{bound}} - f_0)/(\sum_{j=1}^{Q}(a_j/(1 - 2^{-P_j}))) > 0$. Then, the condition $Measure \leq M_{\text{bound}}$ is satisfied. Furthermore, there does not exist any other solution for FB_k, which can result in either a lower error measure (*Measure*) with a lower hardware cost $\left(\sum_{k=1}^{Q} a_k FB_k\right)$, or a lower *Measure* with the same hardware cost, or the same *Measure* with a lower hardware cost, compared to the solution obtained by (10).

Proof: The proof of Theorem 1 consists of four parts. First, it must be shown that given M_{bound}, and all the values of P_k and f_k in (9), there exists a unique FB_k ($k = 1, \ldots, Q$) that can satisfy the condition in (10). In order to prove this, first we show that there exists at least one FB_k that satisfies the condition in (10), then we prove that there is only one unique FB_k that satisfies the condition. Next, we show that using the condition in (10) to set the FB values, it is not possible to obtain an error measure (*Measure*), which is higher than M_{bound} (the maximum possible value of Measure is equal to M_{bound}). Finally, we prove that the last statement of Theorem 1 holds true.

In order to prove the first part, knowing that a_k, f_k, P_k, FB_k, and A are all positive values, we show that when considering two arbitrary consecutive positive integer

FB values, e.g., FB_k and $FB_k + 1$, it is not possible to skip the interval given by (10). In other words, we show that it is not possible to satisfy both the following two conditions at the same time:

$$\frac{Aa_k}{2^{P_k}-1} \geq f_k(2^{-(FB_k+1)})^{P_k},\ f_k(2^{-FB_k})^{P_k} > \frac{Aa_k}{1-2^{-P_k}}.$$

A necessary condition to satisfy both the above inequalities is obtained by combining the two conditions as follows:

$$\frac{Aa_k}{2^{P_k}-1} > 2^{-P_k} \times \frac{Aa_k}{1-2^{-P_k}} \Rightarrow \frac{Aa_k}{2^{P_k}-1} > \frac{Aa_k}{2^{P_k}-1}$$

which does not hold. Hence, there exists at least one value of FB_k that lies within the interval given by (10).

In order to prove the second part, which claims that there exists only one unique value of FB_k that satisfies the condition in (10), let us assume that we have a given value of FB_k that satisfies the condition based on the proof of the first part of the theorem. Also, let j be a nonzero integer. Then, we have two cases.

Case 1) $j < 0$ (FB_k is decreased by $|j|$).

Case 2) $j > 0$ (FB_k is increased by $|j|$).

For Case 1 and according to (10), we have

$$f_k(2^{-FB_k-j})^{P_k} > \frac{Aa_k \times (2^{-j \times P_k})}{2^{P_k}-1} \geq \frac{Aa_k \times 2^{P_k}}{2^{P_k}-1} = \frac{Aa_k}{1-2^{-P_k}}$$

such that (10) does not hold. Similarly, for Case 2, and based on (10) we have

$$f_k(2^{-FB_k-j})^{P_k} \leq \frac{Aa_k \times (2^{-j \times P_k})}{1-2^{-P_k}} \leq \frac{Aa_k \times 2^{-P_k}}{1-2^{-P_k}} = \frac{Aa_k}{2^{P_k}-1}$$

such that (10) does not hold. Thus, if FB_k satisfies the condition in (10), then whenever we increase or decrease FB_k, (10) is no longer satisfied.

In order to prove the third part we notice that according to (10), the maximum value of *Measure* is equal to

$$f_0 + \sum_{k=1}^{Q} \left(A \times \frac{a_k}{1-2^{-P_k}} \right) = M_{\text{bound}}.$$

Hence, by making use of (10) to set all the FB values, we guarantee that the error measure (MM or MSE) does not exceed its maximum bound of M_{bound}.

To prove the final part, i.e., the last statement of Theorem 1, let us define the objective function G as follows:

$$\begin{aligned} G &= A \times hardware\_cost + Measure \\ &= \textstyle\sum_{k=1}^{Q}(Aa_k FB_k + f_k(2^{-FB_k})^{P_k}) \end{aligned} \qquad (11)$$

where $A = (M_{\text{bound}} - f_0)/(\sum_{j=1}^{Q}(a_j/(1-2^{-P_j}))) > 0$. If the solution FB_k in (10) is the global minimum of G in (11), then there does not exist any other solution, e.g., $\widehat{FB}_k$ $k = 1, \ldots, Q$, which can result in either a lower *Measure* with a lower hardware cost, or a lower *Measure* with the same hardware cost, or the same *Measure* with a lower hardware cost, compared to the solution obtained by (10). In fact, if there existed such a solution $\widehat{FB}_k$, then FB_k in (10) would not have been the global minimum of G in (11).

In order to prove the above, note that since A, a_k, f_k, and P_k are all positive values, the functions $Aa_k FB_k$ and $f_k(2^{-FB_k})^{P_k}$ are both convex functions of FB_k. A real-valued function $f(x)$ on an interval is called convex if and only if (iff) for any two points within the interval, e.g., x_1 and x_2, and any $t \in [0, 1]$ we have [51]

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2).$$

Informally, a real-valued function $f(x)$ is convex, if the graph of the function lies below the line segment joining any two points of the graph [51]. Since both $Aa_k FB_k$ and $f_k(2^{-FB_k})^{P_k}$ are convex functions, any positive linear combination of these functions, e.g., G in (11), is also convex [51]. For a convex function, e.g., G in (11), any local minimum is also the global minimum [51]. Therefore, we have to prove that the solution given by (10) is a local minimum of G.

Assume that all the values of FB_k, $k = 1, \ldots, Q$ are set such that the condition in (10) is satisfied. We name the corresponding value of G in (11) for the solution FB_k in (10), as G_{old}. We show that for a small change occurring for an arbitrary FB_k value, i.e., increasing/decreasing FB_k by 1, the new value of G, i.e., G_{new}, cannot be reduced compared to G_{old}. This indicates that the solution in (10), which corresponds to G_{old}, is the local/global minimum of G.

We consider two cases, where one of the FBs (say FB_m, $m \in \{1, \ldots, Q\}$) is reduced/increased by 1, and we show that the new value for G cannot be reduced.

If we reduce FB_m by 1, the new value of G can be obtained as follows:

$$\begin{aligned} G_{\text{new}} = &\sum_{\substack{k=1 \\ k \neq m}}^{Q} (Aa_k FB_k + f_k(2^{-FB_k})^{P_k}) + Aa_m FB_m + \\ &\quad 2^{P_m} f_m(2^{-FB_m})^{P_m} - Aa_m \\ &= \textstyle\sum_{k=1}^{Q}(Aa_k FB_k + f_k(2^{-FB_k})^{P_k}) - Aa_m + \\ &\quad (2^{P_m}-1) f_m(2^{-FB_m})^{P_m} \\ &= G_{\text{old}} - Aa_m + (2^{P_m}-1) f_m(2^{-FB_m})^{P_m} \\ &\Rightarrow G_{\text{new}} - G_{\text{old}} = (2^{P_m}-1) f_m(2^{-FB_m})^{P_m} - Aa_m \\ &\overset{(10)}{\Rightarrow} > Aa_m - Aa_m = 0 \Rightarrow G_{\text{new}} > G_{\text{old}}. \end{aligned}$$

The second case is when FB_m is increased by 1. Under such circumstances, the new value of G is

$$\begin{aligned} G_{\text{new}} = &\sum_{\substack{k=1 \\ k \neq m}}^{Q} (Aa_k FB_k + f_k(2^{-FB_k})^{P_k}) + Aa_m FB_m + \\ &\quad 2^{-P_m} f_m(2^{-FB_m})^{P_m} + Aa_m \\ &= G_{\text{old}} + Aa_m - (1-2^{-P_m}) f_m(2^{-FB_m})^{P_m} \\ &\Rightarrow G_{\text{new}} - G_{\text{old}} = Aa_m - (1-2^{-P_m}) f_m(2^{-FB_m})^{P_m} \\ &\overset{(10)}{\Rightarrow} \geq Aa_m - Aa_m = 0 \Rightarrow G_{\text{new}} \geq G_{\text{old}}. \end{aligned}$$

We can similarly show that if an arbitrary number of FB values are changed by 1, the condition $G_{\text{new}} \geq G_{\text{old}}$ still holds. Hence, the solution given by (10) is the global minimum of G. ■

Theorem 1 provides an analytical approach with a linear complexity to efficiently set the FB of all the intermediate variables in terms of both the error measures MM and MSE and hardware cost, while satisfying the bound on MM (MSE). Theorem 1 can be used for LTI, nonlinear, recursive, and nonrecursive polynomial DFGs. It is also implementable on the top of all the MM and MSE analyses, which make use of

Algorithm 1 Proposed WLO algorithm w.r.t. the error measure MM

```
0- Preprocessing: Factorization
1- Range Analysis By AA
2- Set the FB of the primary input variables and constant
   coefficients [13]
3- Set the FB of intermediate variables: Use Theorem 1
   and (10).
4- MM Computation.
```

linear approximations to represent the error of the DFG with respect to the quantization error of intermediate variables, such as [16] and [37]. Theorem 1 covers both truncation and round-to-nearest types of quantization regarding the error measure MM. However, for MSE optimization, it is only capable of handling the round-to-nearest type of quantization.

The pseudocode of the proposed WLO algorithm for polynomial DFGs with respect to the error measure MM is shown in Algorithm 1. In the preprocessing step (Step 0), the factorization technique in [5] is applied. Step 1 finds suitable IB values for all the variables by performing a range analysis using AA [22]. In Step 2, the FB values of input and constant coefficients are set using AT and the BB search [25] such that a given bound on MM is satisfied. Note that this step returns the values of primary inputs of the polynomial, which also contribute to the MM of the circuit. Furthermore, at this step, all the intermediate variables are assumed to be nonquantized. Note that if the DFG involves feedbacks, AT is paired by IA instead of the BB search to compute MM (Section III-B). Theorem 1 and particularly, (10), is then applied at Step 3 to find the FB of all the intermediate variables. Finally, the BB search in [13] is re-executed with the obtained FB values of intermediate variables to return the final MM in Step 4. Once again, if the DFG involves feedbacks, AT is paired by IA to compute MM at this step.

Note that after we plug the quantization error values of $e_{q\_k} = \pm^{-FB_k-1}$ in the mismatch function y_e in (5), then the particular combination of primary inputs A_j contributing to the value of MM = $\max(|y_e|)$, which has already been computed using the BB search (Step 2), might be changed. However, due to the conditions $|e_{c\_l}| \gg 2^{-FB_k-1}$ and $|A_{e\_j}| \gg 2^{-FB_k-1}$ in Lemma 3, the quantization errors of intermediate variables are set to small values, and as a consequence, the changes that they force upon the main error function y_e are mostly negligible. Hence, it is acceptable to assume that the combination of the primary inputs, which contributes to the MM at the output, does not change when the conditions in Lemma 3 are satisfied. We have not faced such a case in our experiments, where the combination of the primary inputs, which contributes to the MM at the output, changes after performing quantization for intermediate variables, while satisfying the conditions $|e_{c\_l}| \gg 2^{-FB_k-1}$ and $|A_{e\_j}| \gg 2^{-FB_k-1}$ in Lemma 3. Note that even if such a case happens, the MM re-computation at Step 4 shows us that the combination of primary inputs contributing to MM has changed, and hence we can re-execute Step 3 with the new combination of inputs.

The final computed MM always provides safe overestimation since the quantization errors are all assumed to be in their worst-case scenarios (Lemma 1) independently. This might not be the case due to the correlation between the value of primary inputs and the quantization error of intermediate variables.

The WLO algorithm with respect to the error measure MSE is similar to the code in Algorithm 1. In fact, the only difference is that the MSE computation steps correspond to Steps 2 and 4 in Algorithm 1, which should be done based on the discussions in Sections III-C and III-D.

V. Experimental Results

In this section, we compare our algorithms with the previous work. Our work has been implemented in MATLAB, and run on an Intel 2.8 GHz Pentium 4 with 2 GBs of main memory.

The benchmarks used for the experiments, in this section, incorporate both linear and nonlinear circuits with and without feedbacks. The nonlinear benchmarks include a B-Spline function [13], a second-degree polynomial approximation for the arctangent function (*Arctan*) obtained by the least-squares method [26], an efficient degree-8 polynomial approximation of the Gaussian function, i.e., $y = e^{\frac{x^2}{2}}$, presented in [28], a three-variable polynomial of degree 3, *Mibench*, for signal processing, computer graphics and automotive applications [27], a Chebyshev polynomial [13] of degree 9, and an eighth-order LMS filter [64], which is nonlinear and recursive. The linear benchmarks include an 8×8 DCT, a three-variable constant matrix multiplication ($Matrix\_Mult$) for the least-squares-error calibration of three-axis accelerometer sensors [59], where the coefficients are obtained by experimental measurements on the STM32F4 board [60], an 8-K FFT unit from [29], and an eighth-order National Television Systems Committee (NTSC) channel IIR filter from [30], which involves feedbacks. The synthesis tool Xilinx ISE v11 and a CPLD target device have also been chosen for the synthesis process. Note that the range analysis for all the aforementioned benchmarks has been obtained by AA [22].

A. Comparison of AT, AA, and IA for MM Analysis

In this subsection, we present an experimental comparison of the efficiency of AT with AA, IA, and simulation-based methods for the purpose of MM analysis. Note that while IA can be used for range analysis, it is mostly not helpful for computing MM. For instance, assume that the reference output y and its fixed-point realization y_{fixed} lie within the ranges of $(-a, a)$ and $(-b, b)$, respectively, for some positive values of a, b. Using IA alone, the range of error $y_e = y - y_{\text{fixed}}$ is obtained as $(-a - b, a + b)$, which is not realistic. However, IA becomes useful when it is paired with AT (Section III-B).

We have considered six different methods for the purpose of comparison. The first approach is the Gappa tool [58], which uses IA and some user-defined hints to compute MM. The second method uses AA [22], while the third approach uses AT paired with IA, i.e., $AT + IA$, to compute MM (discussion in Section III-B). The fourth method is AT paired with the branch-and-bound search in [25] ($AT + BB$). The fifth method is exhaustive simulations (Exh_Sim) covering all the possible

TABLE I
Comparison of Different MM Analyses

Bench	MM Computation Approach						Runtime (s)			
	Gappa	AA	AT+IA	AT+BB	Rand_Sim	Exh_Sim	Gappa, AA, AT+IA	AT+BB	Rand_Sim	Exh_Sim
B0-Spline	1.334	0.0028	7.63×10^{-4}	7.58×10^{-4}	7.34×10^{-4}	7.42×10^{-4}	< 1	< 10	~ 10	> 500
NTSC	–	1.0321	0.1707		0.1693	–				*Failed*
Matrix_Mult	1.95	3.0186×10^{-4}			2.8×10^{-4}	–				
Mibench	696	0.0344	0.0133		0.008	–				
Arctan	1.242	2.78	2.24×10^{-4}	2.09×10^{-4}	1.95×10^{-4}	2.07×10^{-4}				>500
Gaussian	11.11	0.1063	6.69×10^{-4}	2.92×10^{-4}	2.47×10^{-4}	2.81×10^{-4}				
Chebyshev	4644	108.65	0.2052	0.0062	0.0055	0.006				
Average MM Overestimation (%)	1.46×10^7	4.56×10^5	867.2	**2.59**	*−4.83%*	–	–			

cases, and finally a random simulation approach for 10 000 samples (*Rand_Sim*). Note that *Rand_Sim* is, in general, not a robust way of analyzing MM since it might underestimate the exact value of MM. The results for a number of benchmarks are addressed in Table I. For all the benchmarks in Table I, we have assumed that the primary inputs lie within the normalized range of $[-1, 1]$. The FB of all variables is set to 16 using truncation. The FB of all coefficients is also set to 12 using round-to-nearest. The reference primary inputs are assumed to have the FB of 24.

The MM analyses based on IA, AT, and AA always overestimate the error. However, AT paired with the BB search is the superior static analysis of MM in terms of the amount of overestimations. It can also handle the eighth-order NTSC channel filter, which involves feedbacks due to its linearity. The Gappa tool makes use of IA for the purpose of MM analyses. Hence, it does not deliver tight computations of MM, unless some useful user-defined hints are provided for the tool, which is difficult for larger designs. However, it is interesting to observe that if we enter the AT of the terminal output error, (5), as a user-defined input hint to the Gappa tool, the results become similar to the AT+IA approach in Table I since Gappa uses IA. The exhaustive simulation and Gappa are not applicable to the NTSC filter, as it involves feedbacks. Furthermore, *Exh_Sim* cannot handle the *Mibench* and *Matrix_Mult* benchmarks since the exhaustive search space for those functions involves 2^{24*3} possible cases. AA gives tighter computations of MM compared to IA alone. However, it still constitutes huge overestimations if the degrees of the polynomial and correlations are high. The results in [13] have also shown for some trigonometric functions that the overestimation by AA could be around 1000 times, i.e., around 10^5%. Table I also confirms this for some other benchmarks. The proposed approach AT+IA outperforms AA, and gives almost accurate computations of MM for the first five benchmarks in Table I. However, for the high degree Chebyshev and Gaussian benchmarks, and due to the correlations, the amount of overestimation is still very high (see Table I).

B. *LMS Filter—Nonlinear Recursive DFG*

This subsection addresses experiments for a nonlinear recursive DFG. In particular, we target an eighth-order LMS filter with the DFG presented in [64]. For this filter, we assume the adaptation parameter to be $\mu = 2^{-8} = 0.0039$. Note that it is more efficient in terms of hardware costs to choose μ as powers-of-two [65]. The FB of all variables is set to 16 using truncation. The reference primary input samples $x[k]$ are assumed to have uniform distributions among the interval $[-1, 1]$. The input samples are next added by a Gaussian noise with zero mean and the variance of 10^{-3} [66]. We aim to analyze MM and MSE for this nonlinear recursive filter, where the error is defined as the difference between the quantized and nonquantized LMS filters.

TABLE II
MM Analysis of the Eighth-Order LMS Filter

Approach	MM	# of Loops	Runtime (s)
AA	4.6×10^{-4}	5	6.2212
AT + IA	$\mathbf{2.5201 \times 10^{-4}}$	4	**0.2485**
AT + BB	–	–	*Out of Memory*
Exh_Sim	–	–	*Out of Memory*
Rand_Sim	7.4212×10^{-5}	4	~ 500

In the first experiment, in this subsection, we address the error measure MM. The results of using AA, AT+IA, AT+BB, *Exh_Sim*, and *Rand_Sim* for the purpose of MM analysis are shown in Table II. Note that *Rand_Sim* corresponds to generating 100 000 random samples in this experiment. The *Exh_Sim* method is obviously inapplicable to this design due to the large exhaustive search space. The BB search removes the uncorrelated and linear terms from the search. However, even after this preprocessing, the number of variables is still very high due to the loops, which makes the BB search inapplicable as well. AT+IA gives tighter computations of MM compared to AA. Note that *Rand_Sim* returns an underestimated value of MM, and hence, we do not know the exact number of overestimations provided by AA or AT+IA. It is interesting to observe that although the LMS algorithm requires a high number of iterations to guarantee convergence to the desired time-domain response, the MM analysis between the quantized and nonquantized LMS filters only requires a few number of iterations to guarantee convergence (Column 3). This is also the case for the error measure MSE, which is addressed in the following.

In the second experiment, we evaluate the error measure MSE and compare the results by AT with the approaches in [16] and [37]. We make use of the uniform distribution noise model for the quantization errors in both the AT framework, and the approaches in [16] and [37]. The results are addressed in Table III. We have compared the MSE analyses based on AT in two cases, including and excluding linear approximations (AT and AT_linear in Table III). Both AT and AT_linear give similar MSE values. Compared to the random simulations for 10 000 input samples, AT provides an overestimation of 5.59% when computing MSE, while the MSE analyses in [16] and [37] result in 6.14% of overestimation. The static analyses are all almost accurate.

C. WLO With Respect to the Error Measure MM

In this subsection, we evaluate the efficiency of the proposed WLO algorithm and mainly Theorem 1 with respect to the error measure MM (the code in Algorithm 1).

The first experiment compares the efficiency of Theorem 1 with the near-optimal WLO approach in [61], which is based on derivative computations, as well as a stochastic genetic-algorithm (GA)-based using the *ga*() function in MATLAB with its default options on cross-over, mutation, population size, maximum number of generations, and so on. The WLO results for a number of benchmarks using the error measure MM are addressed in Table IV. Note that we have made use of AT paired with the BB search for the purpose of MM analysis in this experiment since it results in almost accurate computations of MM (see Table I). The primary inputs are all assumed to lie within the normalized range of $[-1, 1]$. Column 2 shows the error bound MM_{bound}.

The GA approach converges to the optimal solution for small benchmarks. However, it fails to find the solution for larger benchmarks since the number of generations exceeds its maximum value (the default value for the maximum number of generations in MATLAB is 100). The methods based on derivative computations [61] and Theorem 1 are both very fast (see Column 5). However, in contrast to Theorem 1, derivative computation does not give the optimal solution in the integer domain (see Columns 3 and 4).

In the second experiment, we evaluate the implementation of the proposed WLO based on Theorem 1 on top different MM analyses. Six optimization methods have been chosen in this experiment. The first approach (M1) is based on [13], which uses a heuristic search algorithm to set the FB of input and coefficients. For the second method (M2), the solution in [13] is enhanced with the polynomial factorization approach in [5]. The third method (M3) is the improvement of M2 with the truncation techniques presented in [14] for only parametric multipliers. Note that the truncation based on [14] is not applicable to constant multipliers and adders. The fourth method (M4) is the implementation of the proposed WLO formula (Theorem 1) on the top of the error analysis based on AA. The fifth approach (M5) implements Theorem 1 on the top of the MM analysis based on AT+IA, and finally the method M6 is the realization of Theorem 1 on the top of the MM analysis based on AT+BB, i.e., AT paired with the BB search.

The results are summarized in Table V, where the primary inputs are all assumed to lie within the normalized range of $[-1, 1]$. The last column addresses the maximum runtime among the approaches M1–M6, including the factorization,

TABLE III
MSE Analysis of the Eighth-Order LMS Filter

Approach	MSE	No. of Loops	Runtime (s)
AT	4.0889×10^{-9}		1.5786
AT_Linear			1.2882
[37], [16]	4.11×10^{-9}	3	4.15
Rand_Sim	3.8724×10^{-9}		3.57

MM computation, and WLO. Note that M6 has the highest runtime among the rest of the methods in Table V. However, regarding the LMS filter (last benchmark in Table V), which is nonlinear and involves feedbacks, M6 goes out of memory according to the BB search. Hence, the worst-case runtime corresponds to M5 regarding the LMS filter benchmark. The high degree Chebyshev and Gaussian polynomials constitute high runtimes due to their time-consuming factorization based on the method in [5]. Note that the portion of the runtimes reported in the last column dedicated to WLO has been less than a second for all the benchmarks. This is due to the simple formula provided by Theorem 1, i.e., (10). As can be observed from the synthesis results, major improvements in terms of area and delay can be obtained when Theorem 1 is implemented on the top of the error analyses based on AT. Note that for the Mibench and Chebyshev functions in Table V, the truncation approach in [14] gives the same answer compared to the WLO formula based on Theorem 1. The reason is that for those benchmarks using our algorithm the truncations are found to be best, when performed only for the parametric multipliers of the DFG. Hence, since the truncation approach in [14] is only applicable to parametric multipliers, the same results have been obtained. However, for other benchmarks, the constant multipliers and adders cannot be truncated using the techniques presented in [14].

The solution based on AA enhanced with our WLO formula (M4) is useful for the cases, where AA does not provide major overestimations of MM. The method based on AT+IA enhanced with Theorem 1 (M5) outperforms M1, M2, and M4. Furthermore, its efficiency is comparable to the superior method M6, as well as M3, both of which require the BB search. Note that M5 becomes more efficient compared to M6 when dealing with nonlinear recursive designs, where the BB search becomes inapplicable (LMS filter in Table V).

In the final experiment, in this subsection, we aim to show that round-to-nearest and truncation can both result in efficient implementations in terms of hardware cost. We have done this experiment for one fifth-order and one sixth-order IIR filters. Our WLO formula with respect to the error measure MM ($MM_{\text{bound}} = 0.001$) has been executed for both circuits, considering rounding and truncation types of quantization. The final results are shown in Table VI. While truncation is a better quantization type for the fifth-order filter, regarding the sixth-order one, rounding requires fewer logic gates and results in a lower value of MM compared to truncation.

D. WLO With Respect to the Error Measure MSE

Now, we address the efficiency of Theorem 1 for the purpose of MSE optimization compared to the WLO approach in [48].

TABLE IV

Comparison of Different WLO Approaches

Bench	MM_{bound}	Area (Gates)			Delay (ns)			WLO Runtime (s)	
		GA	Derivative [61]	Theorem 1	GA	Derivative [61]	Theorem 1	GA	Derivative [61] and Theorem 1
B0-Spline	0.001	**3191**	3368	**3191**	**9.8**	**9.8**	**9.8**	25.92	<0.5
Arctan	0.0005	**1025**	1033	**1025**	**7.331**	**7.331**	**7.331**	14.74	
Gaussian	0.0001	*Failed*	14 736	**12 819**	*Failed*	41.14	**40.91**	–	
Chebyshev	0.002	*Failed*	19 412	**16 355**	*Failed*	34.026	**26.99**	–	
Average Saving With Respect to [61]		–	–	**8.65%**	–	–	**5.31%**	–	

TABLE V

Comparison of Different MM Optimization Algorithms

Bench	MM_{bound}	Area (Gates)						Delay (ns)						Runtime
		M1	M2	M3	M4	M5	M6	M1	M2	M3	M4	M5	M6	
B0-Spline	0.001	8231	6200	4076	3558	**3191**		19.4	14	**9.8**				4.4 s
8 × 8 DCT	0.05	5480		5088	**3760**			6.31		6.242	**6.105**			4.3 s
Arctan	0.0005	2203			9103	1033	**1025**	8.82			14.201	**7.331**		5.12 s
Gaussian	0.0001	67 510		40 198	27 657	16 839	**12 819**	54.465		50.499	47.866	41.14	**40.91**	36.5 s
Mibench	0.003	14 517	10 004	**7159**	8284	**7159**		20.22	15.78	**12.98**	13.219	**12.98**		1.97 s
Chebyshev	0.002	70 628	65 271	**16 355**	43 924	27 449	**16 355**	41.73	44.28	**26.99**	39.79	34.25	**26.99**	49.2 s
LMS	0.004	–			26 273	**25 286**	–	*Failed*			15.142	**15.12**	*Failed*	12.1 s
Average Saving With Respect to M1		–	10.5%	37.6%	14.2%	**55.4%**	**59.1%**	–	7.2%	21.4%	7.05%	**24.5%**	**27.6%**	–

TABLE VI

Truncation Versus Rounding

Filter order	Quantization	MM	#of Gates
5	Trunc	**0.00089346**	**22 426**
	Round	0.00096364	22 585
6	Trunc	0.00093368	26 241
	Round	**0.00080082**	**24 753**

TABLE VII

Comparison of the Proposed FB Optimization Based on Theorem 1 With Respect to the Error Measure MSE With the Approach in [48] on the 8K FFT Unit Benchmark in [29]

Parameter	WLO Approach for the FFT in [29]	
	[48]	Theorem 1 (Proposed)
Runtime (s)	~20	
MSE (dB)	**~26.6**	
Critical Path (ns)	**8.774**	
Total No. of Memory Cells in I/O and Pipeline Stages	404 672	**364 992**
Input/Coef Bit-Width	8/11	
Word-Lengths	11/13/15/16/17	15/15/15/16/12

As an example for the bit-width optimization, we consider the bit-widths of variables and constants in the 8K FFT unit in [29] for digital video broadcasting terrestrial receiver systems [57]. Note that implementing large FFT units on FPGAs requires a huge amount of available memory on the device. Since FPGAs are not as flexible as ASIC designs in terms of having arbitrary large memory units available, we aim to optimize the bit-widths of the variables such that the total number of memory cells, required for the realization of the memories in the intermediate pipeline stages of the FFT unit, and the MSE of the design are both minimized. The analytical bit-width optimization in [48], which is only applicable to LTI circuits, is used to optimize the bit-widths of the FFT unit in [29]. The solution in [48] represents the hardware cost as the total sum of bit-widths, and aims to minimize this value. While the total sum of bit-widths is mostly acceptable to indicate the number of logic gates in an LTI circuit, it cannot illustrate the memory unit sizes used in the FFT architecture efficiently. In fact, a better representation is when we add weights to the bit-widths. For instance, assume that FB_1 is the data-bus bit-width of a large memory with 1024 address lines. Hence, we can set the coefficient a_1 to 1024 such that $a_1 FB_1$ represents the number of memory cells required for the memory. The proposed optimization based on Theorem 1 characterizes the hardware cost as $\sum_{k=1}^{Q} a_k FB_k$, and hence, it results in a more efficient implementation in terms of the total number of memory cells required for the FFT architecture, compared to the other two cases as addressed in Table VII. The last row of Table VII shows the word-lengths (including both IB and FB) of the intermediate pipeline stages realizing the 8K FFT unit, where each pipeline stage stores its results in a memory unit. The address space of the intermediate memories increases exponentially as we move toward the last pipeline stage.

VI. Conclusion and Future Work

In this paper, we proposed pure analytical analyses of precision based on AT for fixed-point recursive and nonrecursive polynomial DFGs in terms of MM, and MSE/SQNR. The analyses did not rely on Monte Carlo simulations. Furthermore, we proposed a WLO formula with linear complexity (Theorem 1), which was implementable on the top of different error analyses, such as AT paired with IA (this paper), AT paired with the branch-and-bound search [25], AA [16], [22]. The optimization goal was to efficiently set the FB of variables, while satisfying the error bounds, including MM, MSE, or SQNR. The efficiency of the proposed analytical WLO approach was also compared with previous work.

As our solutions can only handle fixed-point polynomials, the extension to handle floating-point designs should be developed next. Venturing in this direction is particularly important, as block floating-point designs, which provided a tradeoff between the high accuracy of floating-point circuits and the low hardware cost of fixed-point designs, also gained interest in some DSP applications targeting ASIC/FPGA designs [39]. Moreover, dedicated block floating-point units were recently added to the Altera FPGAs [42], which also indicated their importance. Finally, our analyses could be further extended to cover DFGs with nonarithmetic units such as comparators and if-statements [49].

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions and J. Tong for his help in doing the experiments.

References

[1] N. Tew, P. Kalla, N. Shekhar, and S. Gopalakrishnan, "Verification of arithmetic datapaths using polynomial function models and congruence solving," in *Proc. IEEE/ACM ICCAD*, Nov. 2008, pp. 122–128.

[2] V. J. Mathews and G. L. Sicuranza, *Polynomial Signal Processing*. New York: Wiley-Interscience, 2000.

[3] B. Alizadeh and M. Fujita, "Modular-HED: A canonical decision diagram for modular equivalence verification of polynomial functions," in *Proc. 5th Workshop Constraints Formal Verification*, 2008, pp. 22–40.

[4] D. Gomez-Prado, Q. Ren, M. Ciesielski, J. Guillot, and E. Boutillon, "Optimizing data flow graphs to minimize hardware implementations," in *Proc. IEEE DATE*, Apr. 2009, pp. 117–122.

[5] O. Sarbishei, B. Alizadeh, and M. Fujita, "Polynomial datapath optimization using partitioning and compensation heuristics," in *Proc. IEEE DAC*, Jun. 2009, pp. 931–936.

[6] D. Menard, R. Rocher, and O. Sentieys, "Analytical fixed-point accuracy evaluation in linear time-invariant systems," *IEEE Trans. Circuits Syst. I*, vol. 5, no. 10, pp. 3197–3208, Nov. 2008.

[7] S. Kim, K. Kum, and S. Wonyong, "Fixed-point optimization utility for C and C++ based digital signal processing programs," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 45, no. 11, pp. 1455–1464, Nov. 1998.

[8] H. Keding, F. Hurtgen, M. Willems, and M. Coors, "Transformation of floating-point into fixed-point algorithms by interpolation applying a statistical approach," in *Proc. ICSPAT*, 1998, pp. 270–276.

[9] K. Kum and W. Sung, "Word-length optimization for high level synthesis of digital signal processing systems," in *Proc. IEEE Workshop Signal Processing Syst.*, Oct. 1998, pp. 142–151.

[10] G. Constantinides, P. Cheung, and W. Luk, "Word-length optimization for linear digital signal processing," *IEEE Trans. Comput.-Aided Des.*, vol. 22, no. 10, pp. 1432–1442, Oct. 2003.

[11] O. Sarbishei, Y. Pang, and K. Radecka, "Analysis of range and precision for fixed-point linear arithmetic circuits with feedbacks," in *Proc. IEEE HLDVT*, Jun. 2010, pp. 25–32.

[12] J. Carletta, R. Veillette, F. Krach, and Z. Fang, "Determining appropriate precisions for signals in fixed-point IIR filters," in *Proc. Des. Autom. Conf.*, Jun. 2003, pp. 656–661.

[13] Y. Pang, K. Radecka, and Z. Zilic, "Optimization of imprecise circuits represented by Taylor series and real-valued polynomials," *IEEE Trans. Comput.-Aided Des.*, vol. 29, no. 8, pp. 1177–1190, Aug. 2010.

[14] O. Sarbishei and K. Radecka, "Analysis of precision for scaling the intermediate variables in fixed-point arithmetic circuits," in *Proc. IEEE ICCAD*, Nov. 2010, pp. 739–745.

[15] O. Sarbishei and K. Radecka, "Analysis of mean-square-error (MSE) for fixed-point FFT units," in *Proc. ISCAS*, 2011, pp. 1732–1735.

[16] G. Caffarena, C. Carreras, J. Lopez, and A. Fernandez, "SQNR estimation of fixed-point DSP algorithms," *EURASIP J. Advances Signal Process.*, vol. 2010, pp. 1–11, May 2010.

[17] C. Shi and R. Brodersen, "Automated fixed-point data-type optimization tool for signal processing and communication systems," in *Proc. IEEE DAC*, Jun. 2004, pp. 478–483.

[18] K. Kum and W. Sung, "Combined word-length optimization and highlevel synthesis of digital signal processing systems," *IEEE Trans. Comput.-Aided Des.*, vol. 20, no. 8, pp. 921–930, Aug. 2001.

[19] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee, "Precision and error analysis of MATLAB applications during automated synthesis for FPGAs," in *Proc. DATE*, 2001, pp. 722–728.

[20] A. Gaffar, O. Mencer, W. Luk, and P. Cheung, "Unifying bit-width optimization for fixed-point and floating-point designs," in *Proc. IEEE Symp. Field-Programmable Custom Comput.*, Mar. 2004, pp. 79–88.

[21] W. Sung, "Optimization of number representation," in *Handbook of Signal Processing Systems*, S. S. Bhattacharyya, Ed. Berlin, Germany: Springer, 2010.

[22] D. Lee, A. A. Gaffar, Ray C. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Trans. Comput.-Aided Des.*, vol. 25, no. 10, pp. 1990–2000, Oct. 2006.

[23] A. B. Kinsman and N. Nicolici, "Bit-width allocation for hardware accelerators for scientific computing using SAT-modulo theory," *IEEE Trans. Comput.-Aided Des.*, vol. 29, no. 3, pp. 405–413, Mar. 2010.

[24] Y. Pang, K. Radecka, and Z. Zilic, "An efficient hybrid engine to perform range analysis and allocate integer bit-widths for arithmetic circuits," in *Proc. IEEE ASP-DAC*, Jan. 2011, pp. 455–460.

[25] K. Radecka and Z. Zilic, "Arithmetic transforms for compositions of sequential and imprecise datapaths," *IEEE Trans. Comput.-Aided Des.*, vol. 25, no. 7, pp. 1382–1391, Jul. 2006.

[26] P. N. Baumann. *Arctan(x) Using CORDIC* [Online]. Available: http://www.convict.lu/Jeunes/Math/arctan.htm

[27] M. R. Guthaus, J. S. Ringenberg, and D. Ernst, "Mibench: A free commercially representative embedded benchmark suite," in *Proc. IEEE Workshop Workload Characterization*, Dec. 2001, pp. 3–14.

[28] Y. Chen and N. C. Beaulieu, "A simple polynomial approximation to the Gaussian Q-function and its applications," *IEEE Commun. Lett.*, vol. 13, no. 2, pp. 124–126, Feb. 2009.

[29] R. M. Jiang, "An area-efficient FFT architecture for OFDM digital video broadcasting," *IEEE Trans. Consumer Electron.*, vol. 53, no. 4, pp. 1322–1326, Nov. 2007.

[30] J. Radecki, J. Konard, and E. Dubois, "Design of multidimensional finite-wordlength FIR and IIR filters by simulated annealing," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 42, no. 6, pp. 424–431, Jun. 1995.

[31] D. Menard and O. Sentieys, "Automatic evaluation of the accuracy of fixed-point algorithms," in *Proc. IEEE DATE*, Mar. 2002, pp. 529–535.

[32] D. Menard, R. Rocher, P. Scalart, and O. Sentieys, "SQNR determination in nonlinear and nonrecursive fixed-point systems," in *Proc. Eur. Signal Process.*, 2002, pp. 1349–1352.

[33] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Truncation noise in fixed-point SFGs," *Electron. Lett.*, vol. 35, no. 23, pp. 2012–2014, Nov. 1999.

[34] G. A. Constantinides, "Perturbation analysis for word-length optimization," in *Proc. IEEE Symp. Field-Programmable Custom Comput. Mach.*, Apr. 2003, pp. 81–90.

[35] G. Li and Z. Zhao, "On the generalized DFIIt structure and its state-space realization in digital filter implementation," *IEEE Trans. Circuits Syst.*, vol. 51, no. 4, pp. 769–778, Apr. 2004.

[36] J. A. López, G. Caffarena, C. Carreras, and O. Nieto-Taladriz, "Fast and accurate computation of round-off noise of LTI systems," *IET Circuits, Devices Syst.*, vol. 2, no. 4, pp. 393–408, 2008.

[37] R. Rocher, D. Menard, O. Sentieys, and P. Scalart, "Analytical accuracy evaluation of fixed-point systems," in *Proc. EUSIPCO*, Sep. 2007, pp. 999–1003.

[38] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[39] A. Lipchin, I. Reyzin, D. Lisin, and M. Saptharishi, "Multi-operand block floating-point arithmetic for image processing," in *Proc. IEEE SIPS*, Oct. 2010, pp. 122–127.

[40] O. Sarbishei and K. Radecka, "On the fixed-point accuracy analysis and optimization of FFT units with CORDIC multipliers," in *Proc. IEEE Comput. Arithmetic Conf.*, Jul. 2011, pp. 62–69.

[41] Y. Pang, O. Sarbishei, K. Radecka, and Z. Zilic, "Challenges in verifying and optimizing fixed-point arithmetic-intensive designs," in *Proc. IEEE AQTR*, May 2010, pp. 1–6.

[42] Altera Corporation. (2005, Oct.). *FFT/IFFT Block Floating Point Scaling*, Application Note AN404, version 1.0 [Online]. Available: http://www.altera.com/literature/an/an404.pdf

[43] S. S. Kidambi and P. El-Guibaly, "Area-efficient multipliers for digital signal processing applications," *IEEE Trans. Circuits Systems II: Analog Digit. Signal Process.*, vol. 43, no. 2, pp. 90–95, Feb. 1996.

[44] E. E. Swartzlander, Jr., "Truncated multiplication with approximate rounding," in *Proc. Rec. Asilomar Conf. Signals Syst. Comput.*, vol. 2. 1999, pp. 1480–1483.

[45] L. D. Van and C. C. Yang, "Generalized low-error area-efficient fixed-width multipliers," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 52, no. 8, pp. 1608–1619, Aug. 2005.

[46] C. F. Fang, R. Rutenbar, and T. Chen, "Fast, accurate static analysis for fixed-point finite-precision effects in DSP designs," in *Proc. IEEE ICCAD*, Nov. 2003, pp. 275–282.

[47] D. U. Lee, A. A. Gaffar, O. Mencer, and W. Luk, "Minibit: Bit-width optimization via affine arithmetic," in *Proc. DAC*, Jun. 2005, pp. 837–840.

[48] O. Sarbishei, K. Radecka, and Z. Zilic, "Analytical optimization of bit-widths in fixed-point LTI systems," *IEEE Trans. Comput.-Aided Des.*, vol. 31, no. 3, pp. 343–355, Mar. 2012.

[49] O. Sarbishei and K. Radecka, "Verification of fixed-point datapaths with comparator units using constrained arithmetic transform (CAT)," in *Proc. IEEE ISCAS*, May 2012, pp. 592–595.

[50] O. Sarbishei and K. Radecka, "Fixed-point accuracy analysis of datapaths with mixed CORDIC and polynomial computations," in *Proc. IEEE ASP-DAC*, Jan. 2012, pp. 789–794.

[51] S. Boyd and L. Vandenberghe, *Convex Optimization*, 7th ed. Cambridge, MA: Cambridge Univ. Press, 2009.

[52] O. Sarbishei, M. Tabandeh, B. Alizadeh, and M. Fujita, "High-level optimization of integer multipliers over a finite bit-width with verification capabilities," in *Proc. MEMOCODE*, Jul. 2009, pp. 56–65.

[53] C. C. Wang, C. Shi, R. W. Brodersen, and D. Markovic, "An automatic fixed-point optimization tool in MATLAB XSG/SynDSP environment," *ISRN Signal Processing J.*, vol. 2011, no. 414293, p. 17, Jan. 2011.

[54] B. Alizadeh and M. Fujita, "Modular datapath optimization and verification based on modular-HED," *IEEE Trans. Comput.-Aided Des.*, vol. 29, no. 9, pp. 1422–1435, Sep. 2010.

[55] L. B. Jackson, "On the interaction of round-off noise and dynamic range in digital filters," *Bell Syst. Tech. J.*, vol. 49, pp. 159–184, Feb. 1970.

[56] J. A. López, C. Carreras, and O. Nieto-Taladriz, "Improved interval-based characterization of fixed-point LTI systems with feedback loops," *IEEE Trans. Comput.-Aided Des.*, vol. 26, no. 11, pp. 1923–1933, Nov. 2007.

[57] European Broadcasting Union, "Digital video broadcasting (DVB): Framing structure, channel coding and modulation for digital terrestrial television," *Datasheet: ETSI EN 300 744*, Jan. 2001.

[58] G. Melquiond. (2009, Sep.). *Gappa* [Online]. Available: http://gappa.gforge.inria.fr/

[59] STMicroelectronics. (2010, Apr.). *Tilt Measurement Using a Low-g 3-Axis Accelerometer*, application note AN3182 [Online]. Available: http://www.st.com/internet/com/technical_resources/technical_literature/application_note/cd00268887.pdf

[60] STMicroelectronics User Manual UM1472. (2012, Jan.). *STM32F4DISCOVERY, STM32F4 High-Performance Discovery Board* [Online]. Available: http://www.st.com/internet/com/technical_resources/technical_literature/user_manual/dm00039084.pdf

[61] P. D. Fiore and Li Lee, "Closed-form and real-time word-length adaptation," in *Proc. IEEE ICASSP*, Mar. 1999, pp. 1897–1900.

[62] C. Carreras, J. A. López, and O. Nieto-Taladriz, "Bit-width selection for data-path implementations," in *Proc. Int. Symp. Syst. Synthesis*, Nov. 1999, pp. 114–119.

[63] B. Wu, J. Zhu, and F. N. Najm, "Dynamic-range estimation," *IEEE Trans. Comput.-Aided Des.*, vol. 25, no. 9, pp. 1618–1636, Sep. 2006.

[64] P. K. Meher and M. Maheshwari, "A high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm," in *Proc. IEEE ISCAS*, May 2011, pp. 121–124.

[65] S. C. Douglas, Q. Zhu, and K. F. Smith, "A pipelined LMS adaptive FIR filter architecture without adaptation delay," *IEEE Trans. Comput.-Aided Des.*, vol. 46, no. 3, pp. 775–779, Mar. 1998.

[66] L. D. Van and W. S. Feng, "An efficient systolic architecture for the DLMS adaptive filter and its applications," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 48, no. 4, pp. 359–366, Apr. 2001.

Omid Sarbishei received the B.Sc. and M.Sc. degrees in electrical engineering from the Ferdowsi University of Mashhad, Mashhad, Iran, and the Sharif University of Technology, Tehran, Iran, in 2007 and 2009, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada.

He was with the Integrated Circuits Research Center, Sharif University, in 2009, where he was responsible for implementing a digital video broadcasting terrestrial receiver system under the sponsorship of the Islamic Republic of Iran Broadcasting and the Ministry of Science in Iran. He has also been a Teaching and Research Assistant. His current research interests include computer-aided-design, digital signal processing, arithmetic circuits, formal verification, high-level synthesis, and fault-tolerant sensor fusion.

Mr. Sarbishei was a recipient of the prestigious Lorner Trottier, Provost Graduate, and Graduate Excellence fellowships as a Ph.D. Student. He was also a recipient of the Outstanding Teaching Assistant Award in 2012 among the Faculty of Engineering at McGill University.

Katarzyna Radecka (S'00–M'02) received the B.Eng., M.Eng., and Ph.D. degrees from McGill University, Montreal, QC, Canada, in 1995, 1996, and 2003, respectively.

She is currently with the Department of Electrical and Computer Engineering, McGill University. She was with Nortel, Ottawa, ON, Canada, from 1995 to 1996, with Lucent Technologies, Allentown, PA, from 1996 to 1998, and with Concordia University, Montreal, from 2002 to 2007. She has published over 50 articles and has authored the book, *Verification by Error Modeling: Using Testing Methods for Hardware Verification* (Norwell, MA: Kluwer). Her current research interests include arithmetic circuits, verification, and testing of hardware and software.