Determining fixed-point formats for a digital filter implementation using the worst-case peak gain measure

Anastasia Volkova, Thibault Hilaire, Christoph Lauter

Determining Fixed-Point Formats for a Digital Filter Implementation using the Worst-Case Peak Gain measure

Anastasia Volkova, Thibault Hilaire, Christoph Lauter
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, France
Email: first_name.last_name@lip6.fr

***Abstract*—In this article, we focus on the Fixed-Point implementation of Linear Time Invariant (LTI) filters in state-space representation. For that purpose, we give an algorithm to determine the Fixed-Point Formats of all the involved variables (states and outputs). For the sake of generality, the algorithm works in the case of Multiple Inputs Multiple Outputs filters.**

The computational errors in the intermediate steps of the filter evaluation as well as their accumulation over time are fully taken into account. We handle several rounding modes (round to nearest and truncation) for two's-complement-based Fixed-Point Arithmetic. Our approach is fully rigorous in the way that the output Fixed-Point formats are shown to be free of overflows and we do not use any (non-exhaustive) filter simulation steps but proofs.

I. Introduction

The implementation of a digital filter can be considered as a path from filter specifications to a physical implementation (DSP device). This process includes generation of a transfer function, which defines the relation between the inputs and outputs of a filter. Further one needs to choose a filter structure to be eventually implemented, i.e. a computational scheme, a specific order of filter evaluation. Finally, there is software and hardware code generation.

However, on each step various issues arise. First of all, every transfer function must be discretized, which implies the modification of filter. Secondly, while in infinite precision filter structures (e.g. Direct Forms [1], State-Space [2], Wave [3] etc.) are equivalent, they are no longer the same objects in finite precision. Since the computational schemes are different, the round-off errors and consequently quality of structures vary. Finally, software and hardware implementations are performed under constraints, e.g. power consumption, area, output error, etc. And on each step of the filter implementation the degradation due to finite precision must be taken into account in order to produce a reliable implementation.

To unify the above-described process of filter implementation for any Linear Time-Invariant (LTI) filter and provide reliable implementation, we develop an automatized filter generator. Its work-flow is described in the Fig. 1.

In this paper we focus on one of the various steps: the Fixed-Point (FxP) Algorithm generation step and wordlength optimization. On this step we already know target specifications and software implementation constraints. Therefore, during this optimization, we naturally look for a trade-off between area (for hardware implementation) and computational errors. If we take more bits, the implementation cost arises. If we take less bits, there exist a risk of an overflow. Depending on the application, one or even both criteria can be neglected. However, in the general case on each step of the optimization procedure we need to determine *rigorously* the Fixed-Point Formats (FxPF), i.e. the most and the least significant bits positions, for all variables in the filter implementation. These formats must allow no overflow and take into account computational errors.

This work has been sponsored by the *Agence nationale de la recherche* grant ANR-13-INSE-0007-02 MetaLibm.

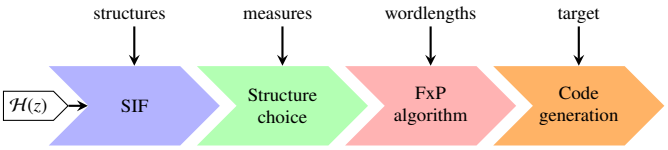

Fig. 1. Automatic filter generator flow.

There exist various approaches on determining FxPF, such as Affine Arithmetic [4], or the common approach of numerous simulations [5]. However, these methods are not suitable for our needs since they do not give any mathematical guarantee on the absence of overflow for any possible input and tend to overestimate. We, on the other hand, propose a rigorous algorithm completely based on mathematical proofs.

In Section II we describe the fixed-point arithmetic in question and theoretical background for our approach. In Section III we define the problem for a filter in a state-space representation and introduce a two-step algorithm. Error analysis of the algorithm is presented in Section IV. Finally, numerical results are provided before conclusion.

Notation: Throughout the article matrices are in uppercase boldface, vectors are in lowercase boldface, scalars are in lowercase. All matrix inequalities and absolute values are considered element-by-element. The matrix $\mathbb{1}$ denotes a matrix of ones with the appropriate size.

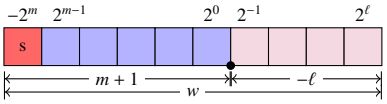

Fig. 2. Fixed-point representation (here, $m = 5$ and $\ell = -4$).

II. Basic bricks

A. The Worst-Case Peak Gain theorem

Let $\mathcal{H} := (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ be a Bounded-Input Bounded-Output stable MIMO LTI filter in state-space representation:

$$\mathcal{H} \begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{cases} \tag{1}$$

where $\boldsymbol{u}(k) \in \mathbb{R}^{q\times 1}$ is the input vector, $\boldsymbol{y}(k) \in \mathbb{R}^{p\times 1}$ the output vector, $\boldsymbol{x}(k) \in \mathbb{R}^{n\times 1}$ the state vector and $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times q}$, $\boldsymbol{C} \in \mathbb{R}^{p\times n}$ and $\boldsymbol{D} \in \mathbb{R}^{p\times q}$ are the state-space matrices.

We will use the following approach on deducing the output interval for a filter: suppose all the inputs are guaranteed to be in a known interval:

$$\forall k, \quad |\boldsymbol{u}_i(k)| \leqslant \bar{\boldsymbol{u}}_i, \quad i = 1, \ldots, q. \tag{2}$$

Then, there exists a N_0 such that $\forall k \geq N_0$ the output $\boldsymbol{y}(k)$ lies in the interval:

$$|\boldsymbol{y}(k)| \leqslant \langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}}, \tag{3}$$

where $\langle\langle \mathcal{H} \rangle\rangle$ denotes the so-called Worst-Case Peak Gain (WCPG) matrix of the filter [6]. It can be computed as the ℓ_1-norm of the impulse response of the filter. The N_0 can be determined using reasoning similar to the WCPG error analysis in [7]. There always exists an input sequence $\boldsymbol{u}(k)$ such that the equality in (3) is reached.

There exist numerous other approaches to determine the output interval, such as involving Affine Arithmetic [4], which may lead to overestimations, or simulations [5], which in turn are inefficient and not rigorous.

It can be shown that the WCPG approach gives the smallest interval containing all the possible values of $\boldsymbol{y}(k)$ and that the bound (3) can be attained. Moreover, in [7] the authors have published an algorithm to evaluate the WCPG at arbitrary precision.

B. Fixed-Point arithmetic for filter implementation

The considered filters are implemented using signed FxP Arithmetic, with two's complement representation [8], [9]. Let t be such a FxP number. It can be written as $t = -2^m t_m + \sum_{i=\ell}^{m-1} 2^i t_i$, where its FxPF (m, ℓ) gives the position of the *most* (MSB) significant and *least* (LSB) significant bits respectively, and $t_i \in \mathbb{B} = \{0, 1\}$ is the i^{th} bit of t as shown in Fig. 2. The wordlength w is given by $w = m - \ell + 1$. The quantization step of the representation is 2^ℓ. Further in the article the LSB position ℓ sometimes will be substituted directly with $m - w + 1$.

The real number t is represented in machine by the integer T, such that $T = t \cdot 2^{-\ell}$, where $T \in [-2^{w-1}; 2^{w-1} - 1] \cap \mathbb{Z}$. Note that this interval is not symmetric.

Let $\boldsymbol{y}(k) \in \mathbb{R}$ be an output of a digital filter. Given the wordlength $\boldsymbol{w}_y$, determining the Fixed-Point Formats for $\boldsymbol{y}(k)$ means to find a MSB vector $\boldsymbol{m}_y$ such that for each k

$$\boldsymbol{y}_i(k) \in [-2^{\boldsymbol{m}_{y_i}}; 2^{\boldsymbol{m}_{y_i}} - 2^{\boldsymbol{m}_{y_i} - \boldsymbol{w}_{y_i} + 1}]. \tag{4}$$

Obviously, we are interested in the least possible MSB, which guarantees (4): any greater MSB would yield to a larger quantization step, hence round-off noise, at constant wordlength.

III. Determining the Fixed-Point Formats

A. Exact filter

The problem of determining the FxPF for a filter $\mathcal{H}$ implementation can be formulated as follows. Let $\mathcal{H}$ be a filter in a state-space representation (1). Suppose all the inputs to be in an interval bounded by $\bar{\boldsymbol{u}}$.

Given the wordlength constraints vector $\boldsymbol{w}_x$ for the state and $\boldsymbol{w}_y$ for the output variables we look for a FxPF for $\boldsymbol{x}(k)$ and $\boldsymbol{y}(k)$ such that for any possible input no overflow occurs. We wish to determine the least MSB vectors $\boldsymbol{m}_y$ and $\boldsymbol{m}_x$ such that

$$\forall k, \quad \boldsymbol{y}(k) \in [-2^{-\boldsymbol{m}_y}; 2^{\boldsymbol{m}_y} - 2^{\boldsymbol{m}_y - \boldsymbol{w}_y + 1}], \tag{5}$$

$$\forall k, \quad \boldsymbol{x}(k) \in [-2^{-\boldsymbol{m}_x}; 2^{\boldsymbol{m}_x} - 2^{\boldsymbol{m}_x - \boldsymbol{w}_x + 1}]. \tag{6}$$

Remark 1. *Since the filter $\mathcal{H}$ is linear and input interval is centered at zero, the output interval is also centered in zero. Therefore, further we will seek to determine the least $\boldsymbol{m}_y$ and $\boldsymbol{m}_x$ such that*

$$\forall k, \quad |\boldsymbol{y}(k)| \leqslant 2^{\boldsymbol{m}_y} - 2^{\boldsymbol{m}_y - \boldsymbol{w}_y + 1}, \tag{7}$$

$$\forall k, \quad |\boldsymbol{x}(k)| \leqslant 2^{\boldsymbol{m}_x} - 2^{\boldsymbol{m}_x - \boldsymbol{w}_x + 1}. \tag{8}$$

B. Applying the WCPG to compute MSB positions

Applying the WCPG theorem on the filter $\mathcal{H}$ yields a bound on the output interval:

$$|\boldsymbol{y}_i(k)| \leqslant (\langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}})_i, \quad i = 1, \ldots, p. \tag{9}$$

Denote the bound vector $\bar{\boldsymbol{y}} := \langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}}$.

We can determine the FxPF for the output of a LTI filter $\mathcal{H}$ utilizing the following lemma.

Lemma 1. *Let $\mathcal{H} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ be a BIBO-stable MIMO LTI filter and $\bar{\boldsymbol{u}}$ be a bound on the input interval. Suppose the wordlengths $\boldsymbol{w}_y$ are known and $\boldsymbol{w}_{y_i} > 1$, $i = 1, \ldots, p$.*

If for $i = 1, \ldots, p$ the MSBs are computed with

$$\boldsymbol{m}_{y_i} = \left\lceil \log_2(\bar{\boldsymbol{y}}_i) - \log_2\left(1 - 2^{1 - \boldsymbol{w}_{y_i}}\right) \right\rceil \tag{10}$$

and the LSBs are computed with $\boldsymbol{\ell}_{y_i} = \boldsymbol{m}_{y_i} + 1 - \boldsymbol{w}_{y_i}$, then for all k $|\boldsymbol{y}_i(k)| \leqslant 2^{\boldsymbol{m}_{y_i}} - 2^{\boldsymbol{m}_{y_i} - \boldsymbol{w}_{y_i} + 1}$ and $\boldsymbol{m}_{y_i}$ is the least.

Proof. We look for the least $\boldsymbol{m}_y$ such that (7) holds. Since the bound $\bar{\boldsymbol{y}}$ can be reached, it is sufficient to require:

$$\bar{\boldsymbol{y}}_i \leqslant 2^{\boldsymbol{m}_{y_i}} - 2^{\boldsymbol{m}_{y_i} - \boldsymbol{w}_{y_i} + 1} \tag{11}$$

Solving this inequality for $\boldsymbol{m}_{y_i}$ we obtain that the smallest integer, which satisfies the above inequality is

$$\boldsymbol{m}_{y_i} = \left\lceil \log_2(\bar{\boldsymbol{y}}_i) - \log_2\left(1 - 2^{1 - \boldsymbol{w}_{y_i}}\right) \right\rceil. \tag{12}$$

■

C. Modification of filter $\mathcal{H}$ to determine bounds on the state variable x

Using Lemma 1 we can determine the FxPF for the output of a filter. In order to determine the FxPF for the state variable we modify the filter $\mathcal{H}$ by vertically concatenating the state vector and the output and include necessary changes into the state matrices.

Denote vector $\boldsymbol{\zeta}(k) := \begin{pmatrix} \boldsymbol{x}(k) \\ \boldsymbol{y}(k) \end{pmatrix}$ to be the new output vector. Then the state-space relationship (1) takes the form:

$$\mathcal{H}_\zeta \begin{cases} \boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{\zeta}(k) = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{C} \end{pmatrix} \boldsymbol{x}(k) + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{D} \end{pmatrix} \boldsymbol{u}(k) \end{cases}. \tag{13}$$

Hence the problem is to find the least MSB vector $\boldsymbol{m}_\zeta$ such that (element-by-element)

$$\forall k, \quad |\boldsymbol{\zeta}(k)| \leqslant 2^{\boldsymbol{m}_\zeta} - 2^{\boldsymbol{m}_\zeta - \boldsymbol{w}_\zeta + 1}. \tag{14}$$

Now, applying the WCPG theorem on the filter $\mathcal{H}_\zeta$ and using Lemma 1, we can deduce the MSB positions of the state and output vectors for an implementation of the filter $\mathcal{H}$.

D. Taking rounding errors into account

However, due to the finite-precision degradation what we actually compute is not the exact filter $\mathcal{H}_\zeta$ but an implemented filter $\mathcal{H}_\zeta^\Diamond$:

$$\mathcal{H}_\zeta^\Diamond \begin{cases} \boldsymbol{x}^\Diamond(k+1) = \Diamond_{\boldsymbol{\ell}_x}\left(\boldsymbol{A}\boldsymbol{x}^\Diamond(k) + \boldsymbol{B}\boldsymbol{u}(k)\right) \\ \boldsymbol{\zeta}^\Diamond(k) = \Diamond_{\boldsymbol{\ell}_\zeta}\left(\begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{C} \end{pmatrix} \boldsymbol{x}^\Diamond(k) + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{D} \end{pmatrix} \boldsymbol{u}(k)\right) \end{cases} \tag{15}$$

where the Sums-of-Products (accumulation of scalar products on the right side) are computed with some rounding operator $\Diamond_\ell$. Suppose, this operator ensures faithful rounding [10]:

$$|\Diamond_\ell(x) - x| < 2^\ell \tag{16}$$

In [11], [12] it was shown that such an operator can be implemented using some extra guard bits for the accumulation.

Denote the errors due to operator $\Diamond_\ell$ as $\boldsymbol{\varepsilon}_x(k)$ and $\boldsymbol{\varepsilon}_y(k)$ for the state and output vectors, respectively. Essentially, the vectors $\boldsymbol{\varepsilon}_x(k)$ and $\boldsymbol{\varepsilon}_y(k)$ may be associated with the noise which is induced by the filter implementation. Then the implemented filter can be rewritten as

$$\mathcal{H}_\zeta^\Diamond \begin{cases} \boldsymbol{x}^\Diamond(k+1) = \boldsymbol{A}\boldsymbol{x}^\Diamond(k) + \boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{\varepsilon}_x(k) \\ \boldsymbol{\zeta}^\Diamond(k) = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{C} \end{pmatrix} \boldsymbol{x}^\Diamond(k) + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{D} \end{pmatrix} \boldsymbol{u}(k) + \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{I} \end{pmatrix} \boldsymbol{\varepsilon}_y(k) \end{cases}, \tag{17}$$

where

$$|\boldsymbol{\varepsilon}_x(k)| < 2^{\boldsymbol{\ell}_x}, \quad \left|\boldsymbol{\varepsilon}_y(k)\right| < 2^{\boldsymbol{\ell}_y}.$$

It should be remarked that since the operator $\Diamond_l$ is applied $\boldsymbol{\varepsilon}_x(k) \neq \boldsymbol{x}(k) - \boldsymbol{x}^\Diamond(k)$ and $\boldsymbol{\varepsilon}_y(k) \neq \boldsymbol{y}(k) - \boldsymbol{y}^\Diamond(k)$. As the rounding also affects the filter state, the $\boldsymbol{x}^\Diamond(k)$ drifts away from $\boldsymbol{x}(k)$ over time, whereas with $\boldsymbol{\varepsilon}_x(k)$ we consider the error due to one step only.

It can be observed that at each instance of time the state and output vectors are computed out of $\boldsymbol{u}(k)$ and error-vectors, which can be considered as inputs as well. Thanks to the linearity of the filters, we can decompose the actually implemented filter into a sum of the exact filter and an "error-filter" $\mathcal{H}_\Delta$ as shown in Fig. 3. Note that this "error-filter" is an artificial one; it is not required to be "implemented" by itself and serves exclusively for error-analysis purposes.

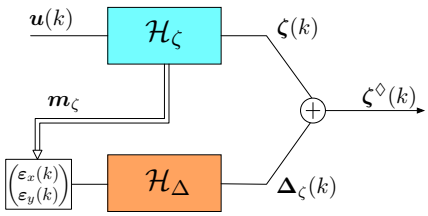

Fig. 3. Implemented filter decomposition.

The filter $\mathcal{H}_\Delta$ is obtained by computing the difference between $\mathcal{H}_\zeta^\Diamond$ and $\mathcal{H}_\zeta$. This filter takes the rounding errors $\boldsymbol{\varepsilon}(k) := \begin{pmatrix} \boldsymbol{\varepsilon}_x(k) \\ \boldsymbol{\varepsilon}_y(k) \end{pmatrix}$ as input and returns the result of their propagation through the filter:

$$\mathcal{H}_\Delta \begin{cases} \boldsymbol{\Delta}_x(k+1) = \boldsymbol{A}\boldsymbol{\Delta}_x(k) + \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{0} \end{pmatrix} \boldsymbol{\varepsilon}(k) \\ \boldsymbol{\Delta}_\zeta(k) = \begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{C} \end{pmatrix} \boldsymbol{\Delta}_x(k) + \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \boldsymbol{\varepsilon}(k) \end{cases}, \tag{18}$$

where, the vector $\boldsymbol{\varepsilon}(k)$ is guaranteed to be in the interval bounded by $\bar{\boldsymbol{\varepsilon}} := 2^{\boldsymbol{\ell}_\zeta}$.

Once the decomposition is done, we can apply the WCPG theorem on the "error-filter" $\mathcal{H}_\Delta$ and deduce the output interval of the computational errors propagated through filter:

$$\forall k, \quad \left|\boldsymbol{\Delta}_\zeta(k)\right| \leqslant \langle\langle \mathcal{H}_\Delta \rangle\rangle \cdot \bar{\boldsymbol{\varepsilon}}. \tag{19}$$

Hence, the output of the implemented filter is bounded with

$$\left|\boldsymbol{\zeta}^\Diamond(k)\right| \leqslant |\boldsymbol{\zeta}(k)| + \left|\boldsymbol{\Delta}_\zeta(k)\right|. \tag{20}$$

Applying Lemma 1 on the implemented filter and using (20) we obtain that the MSB vector $\boldsymbol{m}_\zeta^\Diamond$ can be upper bounded by

$$\begin{aligned} \boldsymbol{m}_{\zeta_i}^\Diamond = \Big\lceil \log_2\Big(\Big(\Big\langle\Big\langle \mathcal{H}_\zeta \Big\rangle\Big\rangle \cdot \bar{\boldsymbol{u}} \Big)_i + (\langle\langle \mathcal{H}_\Delta \rangle\rangle \cdot \bar{\boldsymbol{\varepsilon}})_i \Big) \\ - \log_2\left(1 - 2^{1-\boldsymbol{w}_{\zeta_i}}\right) \Big\rceil. \end{aligned} \tag{21}$$

Therefore, the FxPF $(\boldsymbol{m}_\zeta^\Diamond, \boldsymbol{\ell}_\zeta^\Diamond)$ guarantee that no overflows occur for the implemented filter.

Since the input of the error filter $\mathcal{H}_\Delta$ depends on the FxPF chosen for implementation, we cannot directly use (21). The idea is to first compute the FxPF of the variables in the exact filter $\mathcal{H}$, where computational errors are not taken into account, and use it as an initial guess for implemented filter $\mathcal{H}_\zeta^\Diamond$. Hence, we obtain the following two-step algorithm:

- Step 1: Determine the FxPF $(\boldsymbol{m}_\zeta, \boldsymbol{\ell}_\zeta)$ for the exact filter $\mathcal{H}_\zeta$
- Step 2: Construct the "error-filter" $\mathcal{H}_\Delta$, which gives the propagation of the computational errors induced by format $(\boldsymbol{m}_\zeta, \boldsymbol{\ell}_\zeta)$; then, compute the FxPF $(\boldsymbol{m}_\zeta^\Diamond, \boldsymbol{\ell}_\zeta^\Diamond)$ of the actually implemented filter $\mathcal{H}_\zeta^\Diamond$ using (21).

The above algorithm takes into account the filter implementation errors. However, the algorithm itself is implemented in finite-precision and can suffer from rounding errors, which

influence the output result. All operations in the MSB computation will induce errors, so what we actually compute are only floating-point approximations $\widehat{\boldsymbol{m}_\zeta}$ and $\widehat{\boldsymbol{m}_\zeta}^{\Diamond}$. In what follows, we propose error-analysis of the floating-point evaluation of the MSB positions via (10) and (21).

IV. Error analysis of the MSB computation formula

Let us consider case of $\widehat{\boldsymbol{m}_{\zeta_i}}^{\Diamond}$ and show afterwards that $\widehat{\boldsymbol{m}_{\zeta_i}}$ is its special case. To reduce the size of expressions, denote

$$\mathfrak{m} := \log_2\left(\left(\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle \cdot \bar{\boldsymbol{u}}\right)_i + (\langle\langle \mathcal{H}_\Delta\rangle\rangle \cdot \bar{\boldsymbol{\varepsilon}})_i\right) - \log_2\left(1 - 2^{1-w_{\zeta_i}}\right).$$

Handling floating-point analysis of multiplications and additions in (21) is trivial using approach by Higham [13]. The difficulty comes from the WCPG matrices, which cannot be computed exactly. However both approximations $\widehat{\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle}$ and $\widehat{\langle\langle \mathcal{H}_\Delta\rangle\rangle}$, even if computed with arbitrary precision, bear some errors ε_{WCPG_ζ} and $\varepsilon_{WCPG_\Delta}$ that satisfy

$$0 \leqslant \widehat{\langle\langle \mathcal{H}_\Delta\rangle\rangle} - \langle\langle \mathcal{H}_\Delta\rangle\rangle \leqslant \varepsilon_{WCPG_\zeta} \cdot \mathbb{1} \tag{22}$$

$$0 \leqslant \widehat{\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle} - \left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle \leqslant \varepsilon_{WCPG_\Delta} \cdot \mathbb{1} \tag{23}$$

Introducing the errors on the WCPG computations into the formula (21) we obtain that what we actually compute is

$$\widehat{\boldsymbol{m}_{\zeta_i}}^{\Diamond} \leqslant \left\lceil \mathfrak{m} + \log_2\left(1 + \frac{\varepsilon_{WCPG_\zeta}\sum_{j=1}^{q}\bar{\boldsymbol{u}}_j + \varepsilon_{WCPG_\Delta}\sum_{j=1}^{n+p}\bar{\boldsymbol{\varepsilon}}_j}{\left(\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle \bar{\boldsymbol{u}}\right)_i + (\langle\langle \mathcal{H}_\Delta\rangle\rangle \bar{\boldsymbol{\varepsilon}})_i}\right)\right\rceil. \tag{24}$$

The error term in (24) cannot be zero (apart from trivial case with zero $\bar{\boldsymbol{u}}$). However, since we can control the accuracy of the WCPG matrices, we can deduce conditions for the approximation $\widehat{\boldsymbol{m}_{\zeta_i}}^{\Diamond}$ to be off by at most one.

Lemma 2. *If the WCPG matrices* $\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle$ *and* $\langle\langle \mathcal{H}_\Delta\rangle\rangle$ *are computed such that* (22) *and* (23) *hold with*

$$\varepsilon_{WCPG_\Delta} < \frac{1}{2}\frac{\left(\underline{\langle\langle \mathcal{H}_\Delta\rangle\rangle} \cdot \bar{\boldsymbol{\varepsilon}}\right)_i}{\sum_{j=1}^{p+n}\bar{\boldsymbol{\varepsilon}}_i} \tag{25}$$

$$\varepsilon_{WCPG_\zeta} < \frac{1}{2}\frac{\left(\underline{\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle} \cdot \bar{\boldsymbol{u}}\right)_i}{\sum_{j=1}^{q}\bar{\boldsymbol{u}}_i}, \tag{26}$$

where $\underline{\langle\langle \mathcal{H}\rangle\rangle} := |\boldsymbol{D}| + |\boldsymbol{CB}| + |\boldsymbol{CAB}|$, *then*

$$0 \leqslant \widehat{\boldsymbol{m}_{\zeta_i}}^{\Diamond} - \boldsymbol{m}_{\zeta_i}{}^{\Diamond} \leqslant 1. \tag{27}$$

Proof. Proof by construction, we reason as follows: since the error-term caused by the WCPG floating-point evaluation is positive and the ceil function is increasing, then

$$\widehat{\boldsymbol{m}_{\zeta_i}}^{\Diamond} - \boldsymbol{m}_{\zeta_i}{}^{\Diamond} \geqslant 0, \tag{28}$$

i.e. the floating-point approximation $\widehat{\boldsymbol{m}_{\zeta_i}}^{\Diamond}$ is guaranteed to never be underestimated. However, it can overestimate the MSB position by

$$\widehat{\boldsymbol{m}_{\zeta_i}}^{\Diamond} - \boldsymbol{m}_{\zeta_i}{}^{\Diamond} \leqslant \left\lceil \underbrace{\mathfrak{m} - \lceil \mathfrak{m} \rceil}_{-1<\cdot\leqslant 0} + \log_2\left(1 + \frac{\varepsilon_{WCPG_\zeta}\sum_{j=1}^{q}\bar{\boldsymbol{u}}_j + \varepsilon_{WCPG_\Delta}\sum_{j=1}^{n+p}\bar{\boldsymbol{\varepsilon}}_j}{\left(\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle \bar{\boldsymbol{u}}\right)_i + (\langle\langle \mathcal{H}_\Delta\rangle\rangle \bar{\boldsymbol{\varepsilon}})_i}\right)\right\rceil. \tag{29}$$

The approximation $\widehat{\boldsymbol{m}_{\zeta_i}}^{\Diamond}$ overestimates at most by one bit if and only if the error term is contained in the interval $[0, 1)$, i.e. if

$$0 \leqslant \log_2\left(1 + \frac{\varepsilon_{WCPG_\zeta}\sum_{j=1}^{q}\bar{\boldsymbol{u}}_j + \varepsilon_{WCPG_\Delta}\sum_{j=1}^{n+p}\bar{\boldsymbol{\varepsilon}}_j}{\left(\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle \bar{\boldsymbol{u}}\right)_i + (\langle\langle \mathcal{H}_\Delta\rangle\rangle \bar{\boldsymbol{\varepsilon}})_i}\right) < 1. \tag{30}$$

Hence, using the above condition we can deduce the upper bounds on the ε_{WCPG_ζ} and $\varepsilon_{WCPG_\Delta}$:

$$0 \leqslant \frac{\varepsilon_{WCPG_\zeta}\sum_{j=1}^{q}\bar{\boldsymbol{u}}_j + \varepsilon_{WCPG_\Delta}\sum_{j=1}^{n+p}\bar{\boldsymbol{\varepsilon}}_j}{\left(\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle \bar{\boldsymbol{u}}\right)_i + (\langle\langle \mathcal{H}_\Delta\rangle\rangle \bar{\boldsymbol{\varepsilon}})_i} < 1. \tag{31}$$

Since all the terms are positive, the left inequality is always true. The right inequality in (31) is satisfied for instance if

$$\frac{\varepsilon_{WCPG_\zeta}\sum_{j=1}^{q}\bar{\boldsymbol{u}}_j}{\left(\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle \cdot \bar{\boldsymbol{u}}\right)_i} < \frac{1}{2} \qquad \frac{\varepsilon_{WCPG_\Delta}\sum_{j=1}^{n+p}\bar{\boldsymbol{\varepsilon}}_j}{(\langle\langle \mathcal{H}_\Delta\rangle\rangle \cdot \bar{\boldsymbol{\varepsilon}})_i} < \frac{1}{2}. \tag{32}$$

Rearranging terms we obtain following inequalities on the WCPG computation with error:

$$\varepsilon_{WCPG_\zeta} < \frac{1}{2}\cdot\frac{\left(\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle \bar{\boldsymbol{u}}\right)_i}{\sum_{j=1}^{q}\bar{\boldsymbol{u}}_j} \qquad \varepsilon_{WCPG_\Delta} < \frac{1}{2}\cdot\frac{(\langle\langle \mathcal{H}_\Delta\rangle\rangle \bar{\boldsymbol{\varepsilon}})_i}{\sum_{j=1}^{n+p}\bar{\boldsymbol{\varepsilon}}_j}. \tag{33}$$

Unfortunately, the above results cannot be used in practice, since they depend themselves on the exact WCPG matrices.

It can be shown that $\underline{\langle\langle \mathcal{H}\rangle\rangle}$ is a lower bound of the WCPG matrix. We can compute this matrix exactly. Obviously,

$$\frac{\left(\underline{\langle\langle \mathcal{H}_\Delta\rangle\rangle} \cdot \bar{\boldsymbol{\varepsilon}}\right)_i}{\sum_{j=1}^{p+n}\bar{\boldsymbol{\varepsilon}}_i} \leqslant \frac{(\langle\langle \mathcal{H}_\Delta\rangle\rangle \cdot \bar{\boldsymbol{\varepsilon}})_i}{\sum_{j=1}^{p+n}\bar{\boldsymbol{\varepsilon}}_i} \tag{34}$$

$$\frac{\left(\underline{\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle} \cdot \bar{\boldsymbol{u}}\right)_i}{\sum_{j=1}^{q}\bar{\boldsymbol{u}}_i} \leqslant \frac{\left(\left\langle\left\langle \mathcal{H}_\zeta \right\rangle\right\rangle \cdot \bar{\boldsymbol{u}}\right)_i}{\sum_{j=1}^{q}\bar{\boldsymbol{u}}_i}. \tag{35}$$

Hence, if the WCPG matrices in the right sides of (33) are substituted with their lower bounds, the condition (31) stays satisfied and we obtain bounds (25) and (26).

■

Analogously, Lemma 2 can be applied to the computation of $\widehat{\boldsymbol{m}_{\zeta_i}}$ with the terms concerning filter $\mathcal{H}_\Delta$ set to zero.

In order to guarantee the output MSB position to be optimal, an instance of the Table Maker's Dilemma [10] must be solved. This is due to the simplification by using the triangle inequality in (20), which basically requires both filters $\mathcal{H}_\zeta$ and $\mathcal{H}_\Delta$ reach the worst-case bound with the same input sequence.

V. Complete algorithm

The two-step algorithm, presented in subsection III-D takes into account accumulation of computational errors in a filter over time and Lemma 2 presents error-analysis of the MSB position computation procedure. However, one additional fact has not been taken into account.

In most cases the MSB vectors $\widehat{\boldsymbol{m}}_\zeta$ (computed on Step 1) and $\widehat{\boldsymbol{m}}_\zeta^\Diamond$ (computed on Step 2) are the same. However, in some cases they are not, which can happen due to one of the following reasons:

- the accumulated rounding error due to the FxPF $(\widehat{\boldsymbol{m}}_\zeta, \widehat{\boldsymbol{\ell}}_\zeta)$ makes output of the actually implemented filter pass over to the next binade; or
- the floating-point approximation $\widehat{\boldsymbol{m}}_\zeta^\Diamond$ is off by one.

Moreover, if the MSB position after Step 2 of the algorithm is increased, the LSB position moves along and increases the error. Therefore, the modified format must be re-checked. Hence, the FxPF determination algorithm gets transformed into the following three-step procedure:

Step 1: Determine the FxPF $(\widehat{\boldsymbol{m}}_\zeta, \widehat{\boldsymbol{\ell}}_\zeta)$ for the exact filter $\mathcal{H}_\zeta$;

Step 2: Construct the "error-filter" $\mathcal{H}_\Delta$, which shows the propagation of the computational errors induced by format $(\widehat{\boldsymbol{m}}_\zeta, \widehat{\boldsymbol{\ell}}_\zeta)$; then, compute the FxPF $(\widehat{\boldsymbol{m}}_\zeta^\Diamond, \widehat{\boldsymbol{\ell}}_\zeta^\Diamond)$ of the actually implemented filter $\mathcal{H}_\zeta^\Diamond$;

Step 3: If $\widehat{\boldsymbol{m}}_{\zeta_i}^\Diamond == \widehat{\boldsymbol{m}}_{\zeta_i}$, then return $(\widehat{\boldsymbol{m}}_\zeta^\Diamond, \widehat{\boldsymbol{\ell}}_\zeta^\Diamond)$; otherwise $\widehat{\boldsymbol{m}}_{\zeta_i} \longleftarrow \widehat{\boldsymbol{m}}_{\zeta_i} + 1$ and go to Step 2.

VI. Numerical results

The above described algorithm was implemented as a C library, using GNU MPFR[1] [14] version 3.1.12, GNU MPFI[2] version 1.5.1 and the WCPG library [7].

Consider following example: let $\mathcal{H}$ be a random stable filter with 3 states, 1 input and 1 output. Suppose the inputs are in an interval absolutely bounded by $\bar{\boldsymbol{u}} = 5.125$. In this example we set all the wordlength constraints to 7 bits, however our library supports the multiple wordlength paradigm.

The results of the work of our algorithm can be observed in Table I. On the Step 1 we deduce the MSB positions for the filter $\mathcal{H}$, which does not take computational errors into account. These MSBs serve as an initial guess for the Step 2. We compute the vector $\bar{\boldsymbol{\varepsilon}}_\zeta$ and construct the "error-filter" $\mathcal{H}_\Delta$. Taking into account the error propagation yields changes in the MSB positions: the rounding errors force the third state $\boldsymbol{x}_3(k)$ to pass over to the next binade. However, moving the MSB yields larger quantization step and an addition step is required. Step 3 verifies that the FxPF deduced on the Step 2 guarantees no overflow.

[1] http://www.mpfr.org/

[2] https://gforge.inria.fr/projects/mpfi/

	states			output
	$\boldsymbol{x}_1(k)$	$\boldsymbol{x}_2(k)$	$\boldsymbol{x}_3(k)$	$\boldsymbol{y}(k)$
Step 1	6	7	5	6
Step 2	6	7	6	6
Step 3	6	7	6	6

TABLE I
Evolution of the MSB positions vector through algorithm iterations

We can verify the result by tracing the state and output vectors in two cases: the exact vectors $\boldsymbol{y}(k)$ and $\boldsymbol{x}(k)$; and the quantized vectors $\boldsymbol{y}^\Diamond(k)$ and $\boldsymbol{x}^\Diamond(k)$. The quantized vectors are the result of an implementation of the filter $\mathcal{H}$ with FxPF deduced on the Step 1. For each state and output we take an input sequence which makes the state respectively output attain the theoretical bound. However, this sequence is not guaranteed to maximize the quantized error.

In Fig. 4 we can observe that the exact vector $\boldsymbol{y}(k)$ attains the bound computed with the WCPG theorem. The quantized output stays within the bound $\bar{\boldsymbol{y}}^\Diamond$ for the implemented filter but passes over the bound of the exact filter. However, as the bound $\bar{\boldsymbol{y}}$ is far from the next binade, the MSB position is not changed and the computational errors do not result in overflow.

For the third state $\boldsymbol{x}_3(k)$, its bound $\bar{\boldsymbol{x}}_3$ is very close to the $2^{\boldsymbol{m}_{x_3}} - 2^{\boldsymbol{\ell}_{x_3}}$ and taking into account the filter computational errors yields passing to the next binade. It can be clearly observed in Fig. 5 that the quantized state passes over the value $2^{\boldsymbol{m}_{x_3}} - 2^{\boldsymbol{\ell}_{x_3}}$. If the computational errors were not taken into account, the initial format (deduced on the Step 1) would result in overflow.

Therefore, we observed that deducing the FxPF without taking into account the computational errors can result in overflow but our approach guarantees a reliable implementation without overestimation.

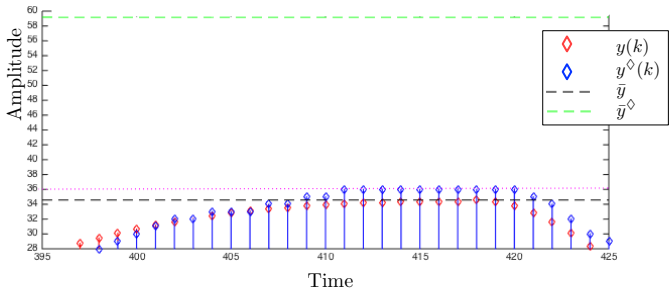

Fig. 4. The exact and quantized outputs of the example. Quantized output does not pass over to the next binade.

VII. Conclusions

We give an algorithm to determine the FxPF for all variables of a LTI filter, where the computation errors are taken into account. We ensure, by construction, that no overflow occurs. However, the computed MSB positions can overestimate at most by one bit. In order to guarantee no overestimation at all an instance of the TMD must be solved. Our algorithm can be extended to any LTI filter realization (Direct Forms, Lattice filters, etc.) using the Specialized Implicit Framework (SIF) tool

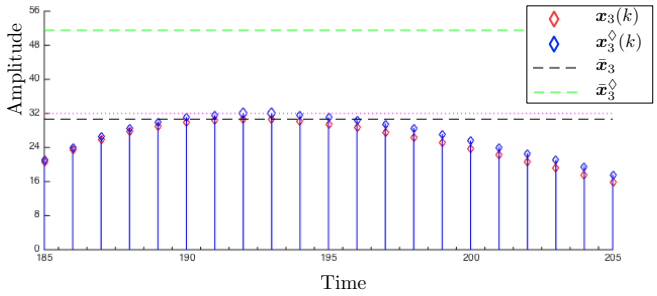

Fig. 5. The exact and quantized third state of the example. Quantized output passes over to the next binade.

[6]. SIF enables unification of various LTI filter realizations for analysis, comparison and FxP code generation. Therefore, this work is an essential basic brick for the automatic filter-to-code conversion.

References

[1] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall,, 1975.

[2] B. Friedland, *Control Systems Design: An Introduction to State-Space Methods*. McGraw-Hill Higher Education, 1985.

[3] A. Fettweiss, "Wave digital filters: Theory and practice," *Proc. of the IEEE*, vol. 74, no. 2, 1986.

[4] J. Lopez, C. Carreras, and O. Nieto-Taladriz, "Improved interval-based characterization of fixed-point LTI systems with feedback loops," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 11, pp. 1923–1933, 2007.

[5] S. Kim, K. Kum, and W. Sung, "Fixed-point optimization utility for C and C++ based digital signal processing programs," *IEEE Transactions on Circuits and Systems*, vol. 45, pp. 1455–1464, November 1998.

[6] T. Hilaire and B. Lopez, "Reliable implementation of linear filters with fixed-point arithmetic," in *Proc. SiPS*, 2013.

[7] A. Volkova, T. Hilaire, and C. Lauter, "Reliable evaluation of the worst-case peak gain matrix in multiple precision," in *Proc. 22nd IEEE Symposium on Computer Arithmetic (ARITH22)*, 2015.

[8] J. v. Newmann, "First draft of a report on the edvac," tech. rep., 1945.

[9] T. Finley, "Two's complement." Cornell University lecture notes, 2000.

[10] J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres, *Handbook of Floating-Point Arithmetic*. Birkhäuser Boston, 2010.

[11] B. Lopez, T. Hilaire, and L. S. Didier, "Formatting bits to better implement signal processing algorithms," in *Proc.PECCS, Portugal*, pp. 104–111, 2014.

[12] F. de Dinechin, M. Istoan, and A. Massouri, "Sum-of-product architectures computing just right," in *Application-Specific Systems, Architectures and Processors (ASAP)*, IEEE, 2014.

[13] N. J. Higham, *Accuracy and stability of numerical algorithms (2. ed.)*. SIAM, 2002.

[14] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann, "MPFR: A multiple-precision binary floating-point library with correct rounding," *ACM Transactions on Mathematical Software*, vol. 33, no. 2, pp. 13:1–13:15, 2007.