Compact Framework for Reproducible Analysis of Finite Wordlength Effects in Linear Digital Networks

David Luengo*, David Osés*, Fernando Cruz-Roldán†

*Dep. of Signal Theory and Communications, Universidad Politécnica de Madrid, 28031 Madrid (Spain)
†Dep. of Signal Theory and Communications, Universidad de Alcalá, 28805 Alcalá de Henares (Spain)

***Abstract*—The analysis of finite wordlength effects in linear digital networks requires specifying the exact order in which all the internal computations are performed, as well as the type and resolution of all the quantizers. Popular digital filter descriptions (difference equations, transfer function, state space representation, etc.) are unable to provide a bit-true description valid for any filter structure, thus preventing most of the results in the literature from being truly reproducible. Furthermore, the quantizers are often not properly described. In this work, we introduce a novel and compact framework to describe unambiguously any single-input single-output (SISO) linear digital network. The proposed approach is simple, efficient and guarantees the reproducibility of the results obtained for any network by allowing us to describe in detail the data flow as well as the quantization of all the coefficients and operations performed. An example of a third order Butterworth filter with four different implementations is provided to show the descriptive power and flexibility of the proposed approach.**

I. Introduction

The reproducibility of the results presented in scientific works by researchers is essential to gauge the performance of the different approaches proposed. For this reason, reproducible research is currently being highly emphasized in many fields like signal processing [1], biostatistics [2], or computational science [3]. In linear digital networks, finite wordlength (FWL) effects occur when these networks are implemented using fixed point arithmetic [4], and many works have been devoted to the analysis of FWL effects. Reproducible analysis of FWL effects in linear digital networks requires specifying the exact order in which all the internal computations are performed, as well as the type and resolution of all the quantizers. However, most works do not contain all the information that an independent researcher would require to reproduce the results presented with little effort.

In this work, we introduce first a compact single matrix representation of the network's structure. This description, which is based on [5], allows us to specify unambiguously the data flow (i.e., the order of all the operations performed by the network). Then, another quantization matrix is introduced to specify all the quantizers (i.e., their type and resolution) used in the network's implementation. Altogether, these two matrices allow us to perform completely reproducible simulations of the error propagation through the network, an issue which is essential to analyze the effect of FWL effects.

The paper is structured as follows. Firstly, in Section II we briefly review some of the proposed alternatives for describing linear digital networks. Then, we describe in detail the proposed framework in Section III and a case study (a third order Butterworth filter with four different implementations) is presented in Section IV. The description of all the implementations is provided and used to analyze their behavior w.r.t. the number of quantization bits used. Finally, the main contributions of this work and some potential future lines are summarized in Section V.

II. Description of Linear Digital Networks

The fixed point implementation of linear digital networks causes the appearance of many undesirable finite wordlength effects: quantization noise, deviation from the desired frequency response due to coefficient sensitivity, zero-input limit cycles, etc. [4]. In order to ensure the reproducibility of the results, the analysis of finite word length (FWL) effects requires specifying the exact order in which all the internal computations are performed, as well as the type and properties of all the quantizers. One of the most popular approaches for describing single-input single-output (SISO) linear digital networks is the *state-space formulation* [6]:

$$\mathbf{v}[n+1] = \mathbf{A}\mathbf{v}[n] + \mathbf{b}x[n], \tag{1}$$

$$y[n] = \mathbf{c}^\top \mathbf{v}[n] + dx[n], \tag{2}$$

where $x[n]$ is the input sequence, $y[n]$ is the output, $\mathbf{v}[n]$ is the $M \times 1$ state vector, $\mathbf{A}$ is the $M \times M$ state matrix, $\mathbf{b}$ is the $M \times 1$ input weight vector, $\mathbf{c}$ is the $M \times 1$ output weight vector, and d is a feedforward constant term.

Unfortunately, the state-space formulation does not provide a precise description of the network's implementation (a given state-space formulation can be implemented using many completely different structures) and is unable to describe structures where the next state depends on the current state of other internal nodes. In order to overcome these limitations, some alternative formulations have been proposed (cf., [4], [6]). One of the most commonly used considers a $P \times 1$ ($P \geq M$) "extended" state vector, $\mathbf{w}[n]$, that includes the internal nodes and the outputs [7]. Making use of $\mathbf{w}[n]$, Eqs. (1) and (2) can be expressed more compactly as

$$\mathbf{w}[n+1] = \mathbf{e}x[n] + \mathbf{F}\mathbf{w}[n+1] + \mathbf{G}\mathbf{w}[n], \tag{3}$$

where $\mathbf{e} \in \mathbb{R}^{P\times 1}$, $\mathbf{F} \in \mathbb{R}^{P\times P}$ and $\mathbf{G} \in \mathbb{R}^{P\times P}$. This formulation allows us to express the dependence of the next state on the values of internal nodes explicitly and is closer to the actual implementation of the filter, but does not result in an automatic implementation yet.

Some additional formulations for the detailed description of generic linear digital network structures have been developed (see, e.g., [8]), but they are all based on several matrices,

ISBN 978-0-9928626-7-1

and thus complex and difficult to use. A more compact single matrix representation, based on Eq. (3) and leading directly to a practical implementation, has been recently proposed [5]. In this paper, we first simplify the network description provided in [5], and then extend it by incorporating a quantization matrix that describes precisely all the quantizers used. These two matrices, altogether with the precise specification of the input, provide a compact and unambiguous representation of any SISO linear digital network that allows us to analyze the effects of FWL effects, as detailed in the following section.

III. Proposed Framework

A. Network Description

First of all, let us remark that $\mathbf{e}$, $\mathbf{F}$ and $\mathbf{G}$ are typically sparse, with non-zero elements only in the positions corresponding to connections among elements in the network. Hence, following the approach of [5], a compact representation can be obtained using a single matrix that contains only the relevant information about those connections. In the sequel, we describe a simplified version of [5], where redundant terms have been removed.

Let us consider a generic SISO linear digital network. This network can only be composed of multipliers, accumulators and delay elements. The proposed approach starts by numbering the different nodes in the following way:

- The first M nodes $(1, \ldots, M)$ are assigned to the outputs of the memory cells (i.e., delay nodes), beginning by the outermost output when several delay nodes are connected in series.
- The following node (i.e., the $(M+1)$-th node) corresponds to the input.
- The final $L = P - (M+1)$ nodes $(M+2, \ldots, P = M+L+1)$ are assigned to the outputs of internal nodes that do not correspond to delay units (i.e., outputs of multipliers or accumulators), starting from those closer to the inputs and proceeding towards the outputs.
- The last node (i.e., the P-th node) is the output node.

Following these simple rules, we can ensure the computability of the resulting network, which can be described compactly by using a single *network matrix*,

$$\mathbf{N} = \begin{bmatrix} \widetilde{\mathbf{G}} \\ \widetilde{\mathbf{F}} \end{bmatrix}, \tag{4}$$

composed of two sub-matrices. First of all, $\widetilde{\mathbf{G}}$, is the $M \times 3$ matrix containing the input-output relationship of the delay elements,

$$\widetilde{\mathbf{G}} = \begin{bmatrix} 1 & i_1 & 1 \\ 2 & i_2 & 1 \\ \vdots & \vdots & \vdots \\ M & i_M & 1 \end{bmatrix}, \tag{5}$$

where the m-th row of $\widetilde{\mathbf{G}}$ $(1 \le m \le M)$ implies that $w_m[n+1] = w_{i_m}[n]$, as shown in Fig. 1(a). Then, $\widetilde{\mathbf{F}}$ is the $K \times 3$ matrix containing the internal connections to non-delay units,

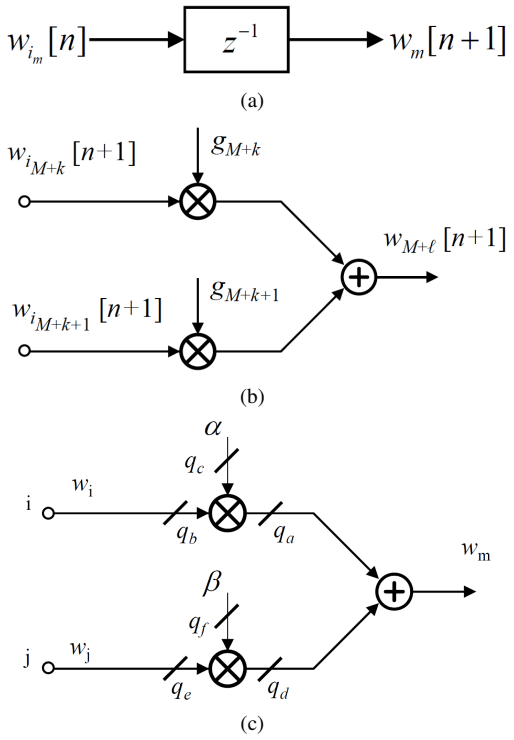

Fig. 1. Basic building blocks of a linear network. (a) Delay element. (b) Multiply and accumulate (MAC) operation. (c) MAC with quantizers included.

$$\widetilde{\mathbf{F}} = \begin{bmatrix} M+2 & i_{M+2} & g_{M+2} \\ \vdots & \vdots & \vdots \\ M+\ell & i_{M+k} & g_{M+k} \\ M+\ell & i_{M+k+1} & g_{M+k+1} \\ \vdots & \vdots & \vdots \\ M+L+1 & i_{M+K+1} & g_{M+K+1} \end{bmatrix}, \tag{6}$$

where the rows in $\widetilde{\mathbf{F}}$ containing the same node in the first column refer to each of the branches in the accumulator used to obtain the value of that node, and $K \ge L$ is the total number of branches in the internal multipliers and accumulators of the linear network. For instance, the two middle rows in (6) with $M+\ell$ in the first column encode the MAC operation shown in Fig. 1(b), whose output is

$$w_{M+\ell}[n+1] = g_{M+k} w_{i_{M+k}}[n+1] + g_{M+k+1} w_{i_{M+k+1}}[n+1].$$

B. Quantizers and Filtering

In order to be able to perform truly reproducible FWL simulations of the network under study, we need to specify all the quantizers used. For this purpose, we introduce the following *quantization matrix*,

$$\mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ \vdots & \vdots & \vdots \\ q_{k1} & q_{k2} & q_{k3} \\ \vdots & \vdots & \vdots \\ q_{K1} & q_{K2} & q_{K3} \end{bmatrix}, \tag{7}$$

ISBN 978-0-9928626-7-1 © EURASIP 2017

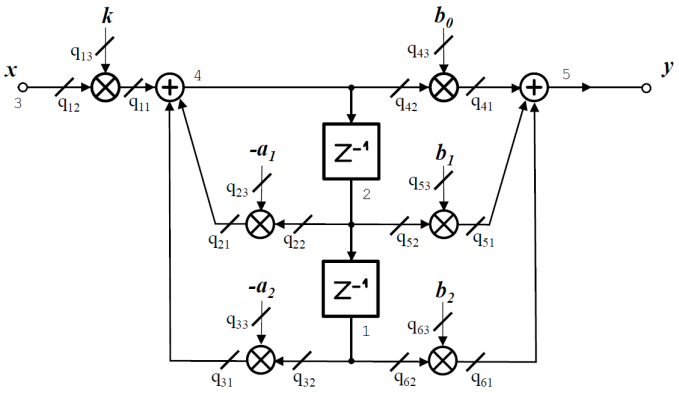

Fig. 2. Second order IIR filter implemented using direct form II.

where the first column specifies the quantizers for the outputs of each of the internal nodes, the second column defines the quantizers used for each of the inputs, and the third column the quantizers used for the coefficients (gains). For instance, the structure represented in Fig. 1(c) corresponds to the following portion of the $\widetilde{\mathbf{F}}$ and $\mathbf{Q}$ matrices.

$$\widetilde{\mathbf{F}} = \begin{bmatrix} \vdots & \vdots & \vdots \\ m & i & \alpha \\ m & j & \beta \\ \vdots & \vdots & \vdots \end{bmatrix} \qquad \mathbf{Q} = \begin{bmatrix} \vdots & \vdots & \vdots \\ q_a & q_b & q_c \\ q_d & q_e & q_f \\ \vdots & \vdots & \vdots \end{bmatrix}.$$

From a mathematical point of view, the output in Fig. 1(c) would be given by

$$w_m[n+1] = q_a(q_b(w_i[n+1])q_c(\alpha)) + q_d(q_e(w_j[n+1])q_f(\beta)).$$

As a second example, consider the classical second order IIR filter implemented using direct form II shown in Fig. 2. The network and quantizer matrices for this network are the following:

$$\widetilde{\mathbf{F}} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 1 \\ 4 & 3 & k \\ 4 & 2 & -a_1 \\ 4 & 1 & -a_2 \\ 5 & 4 & b_0 \\ 5 & 2 & b_1 \\ 5 & 1 & b_2 \end{bmatrix} \qquad \mathbf{Q} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \\ q_{41} & q_{42} & q_{43} \\ q_{51} & q_{52} & q_{53} \\ q_{61} & q_{62} & q_{63} \end{bmatrix}.$$

Note that each of the elements in (7) is not a single number, but a quantizer structure that specifies the quantizer's type and properties. In our implementation of the filters we use the quantizer object generated by Matlab's quantizer function, which allows us to specify precisely many types of quantizers (including the option `'double'` that we use to simulate the absence of a fixed point quantizer) and any desired resolution.[1] Table I shows the Matlab implementation of the filtering function that automatically reads the network and quantization matrices and computes the output for a given input sequence.

[1]Note that, although we have used Matlab for the simulations, the proposed framework is very general and can be easily integrated within any software that allows the use of quantizers.

TABLE I. MATLAB FILTERING IN THE PROPOSED FRAMEWORK.

```
[y, cf] = filterNq(N, x, ci, Q)

% y : Output vector.
% cf : Final state of the network.
% N : Network matrix.
% x : Input vector.
% ci : Initial state of the network.
% Q : Quantizer matrix.

L = max(N(:,1));
w = zeros(L,1);
m = length(find(N(:,1)-N(:,2) < 0));
w([N(1:m,2)]) = ci;

for k = m+1:size(N,1)
    if N(k,3) ~= 1
        N(k,3) = quantize(Q(k-m,3),N(k,3));
    end
end

for n = 1:length(x)
    w(1:m) = w([N(1:m,2)]);
    w(m+1) = x(n);
    w(m+2:end) = 0;
    for k = m+1:size(N,1);
        w(N(k,1)) = w(N(k,1)) + ...
                    quantize(Q(k-m,1),N(k,3) * quantize(Q(k-m,2),w(N(k,2))));
    end
    y(n) = w(end);
end

cf = w(1:m);
```

IV. NUMERICAL SIMULATIONS

In this section we present two case studies. In the first example, we compare four different implementations of the same third-order Butterworth filter, studying their performance as the number of bits is allowed to increase. In the second example, we estimate the frequency response and the noise power spectral density of the FWL implementation of the second-order system studied in the first example of [9] to show that we are able to reproduce their results. In both cases, we use the noise load method proposed in [9] to construct the random input signals that are used to probe the filter's performance. Note that this method was originally applied only to direct form filters, but we have extended it here to work with any desired structure.

A. Performance vs. Resolution

Let us consider the bireciprocal lattice wave digital filter shown in Fig. 3, whose network matrix is given by

$$\mathbf{N}_{Lat} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 7 & 1 \\ 3 & 4 & 1 \\ 5 & 4 & -1 \\ 5 & 1 & 1 \\ 6 & 5 & -1/3 \\ 6 & 1 & -1 \\ 7 & 5 & 1 \\ 7 & 6 & 1 \\ 8 & 3 & 1 \\ 8 & 6 & 1 \\ 9 & 8 & 1/2 \end{bmatrix}.$$

In the sequel, we will investigate the performance of the lattice implementation of the filter and three alternative implementations. More precisely, we consider the direct form I (DF1) and

ISBN 978-0-9928626-7-1 © EURASIP 2017

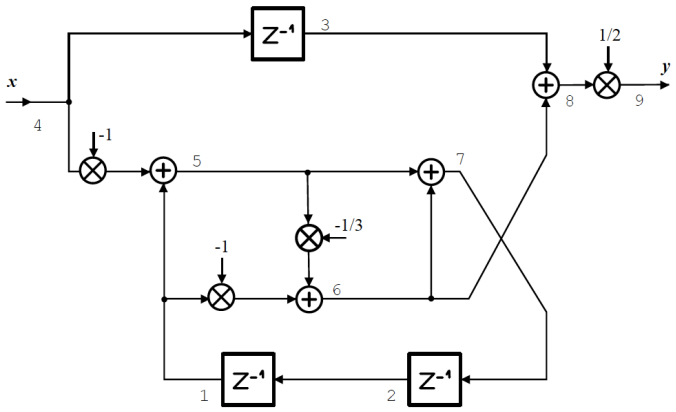

Fig. 3. Lattice implementation of the third order Butterworth filter analyzed in Section IV-A.

direct form II (DF2), whose network matrices are

$$\mathbf{N}_{DF1} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 1 \\ 3 & 7 & 1 \\ 4 & 5 & 1 \\ 5 & 6 & 1 \\ 6 & 8 & 1 \\ 8 & 7 & b_0 \\ 8 & 3 & b_1 \\ 8 & 2 & b_2 \\ 8 & 1 & b_3 \\ 8 & 4 & -a_3 \\ 8 & 5 & -a_2 \\ 8 & 6 & -a_1 \end{bmatrix} \quad \mathbf{N}_{DF2} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 1 \\ 3 & 5 & 1 \\ 5 & 4 & 1 \\ 5 & 3 & -a_1 \\ 5 & 2 & -a_2 \\ 5 & 1 & -a_3 \\ 6 & 5 & b_0 \\ 6 & 3 & b_1 \\ 6 & 2 & b_2 \\ 6 & 1 & b_3 \end{bmatrix},$$

as well as the direct form II transposed (DF2T), whose network matrix is given by

$$\mathbf{N}_{DF2T} = \begin{bmatrix} 1 & 6 & 1 \\ 2 & 7 & 1 \\ 3 & 8 & 1 \\ 5 & 4 & b_0 \\ 5 & 1 & 1 \\ 6 & 4 & b_1 \\ 6 & 2 & 1 \\ 6 & 5 & -a_1 \\ 7 & 4 & b_2 \\ 7 & 3 & 1 \\ 7 & 5 & -a_2 \\ 8 & 4 & b_3 \\ 8 & 5 & -a_3 \\ 9 & 5 & 1 \end{bmatrix}.$$

The coefficients for the three direct forms are obtained through the following Matlab command: `[b,a] = butter(3,0.5)`. This results in $a_1 \approx 0$, $a_2 = 0.3333$, $a_3 \approx 0$, $b_0 = 0.1667$, $b_1 = 0.5000$, $b_2 = 0.5000$ and $b_3 = 0.1667$. Regarding the quantization matrix, the same quantizer type is used for all the inputs, outputs and coefficients: a signed fixed-point quantizer (`mode = 'fixed'`), rounding towards $-\infty$ (`roundmode = 'floor'`), saturating in overflow (`overflowmode = 'saturate'`), using B bits for the fractional part and 3 bits for the integer part (i.e., $B+3$ bits overall). Figure 4 shows the average level of the noise power spectral density (PSD) in dB (`PmdB` in the algorithm of Table II), as a function of the number of bits used for the fractional part ($B = 8, \ldots, 24$) for the four implementations analyzed in this section. The inputs are constructed according to the noise load method described in [9]. Table II shows our Matlab implementation of the noise load method for any structure described using the framework proposed in Section III. Note that the performance of the lattice implementation is the best one as expected, whereas all the direct forms provide very similar results (less than 0.3 dB in difference). Note also the 6 dB increase in performance with each bit added in all the implementations.

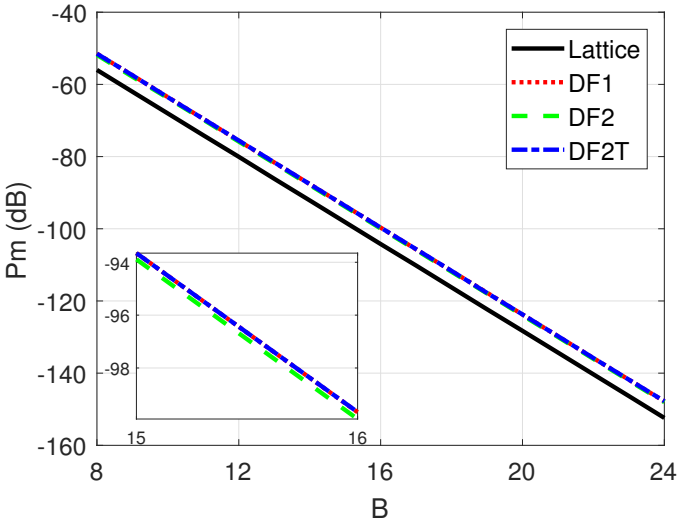

Fig. 4. Average level of the noise power spectral density (PSD) in dB, as a function of the number of bits used for the fractional part ($B = 8, \ldots, 24$) for the four implementations analyzed in Section IV-A.

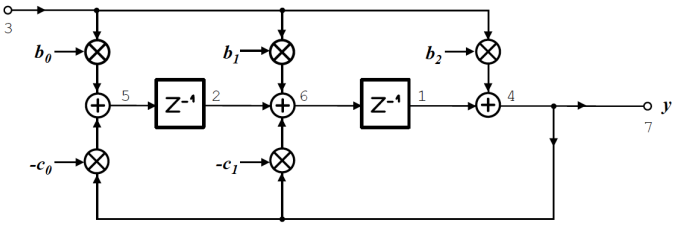

Fig. 5. Implementation of the second-order system (SOS) filter analyzed in Section IV-B.

B. Noise Response

Let us consider the second-order system (SOS) shown in Figure 5. This filter corresponds to the following network matrix and quantizer matrix implementation:

$$\mathbf{N}_{SOS} = \begin{bmatrix} 1 & 6 & 1 \\ 2 & 5 & 1 \\ 4 & 3 & b_2 \\ 4 & 1 & 1 \\ 5 & 3 & b_0 \\ 5 & 4 & -c_0 \\ 6 & 2 & 1 \\ 6 & 3 & b_1 \\ 6 & 4 & -c_1 \\ 7 & 4 & 1 \end{bmatrix}, \quad \mathbf{Q}_{SOS} = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \\ q_{41} & q_{42} & q_{43} \\ q_{51} & q_{52} & q_{53} \\ q_{61} & q_{62} & q_{63} \\ q_{71} & q_{72} & q_{73} \\ q_{81} & q_{82} & q_{83} \end{bmatrix},$$

ISBN 978-0-9928626-7-1 © EURASIP 2017

TABLE II. MATLAB IMPLEMENTATION OF THE EXTENDED NOISE LOAD METHOD ORIGINALLY DESCRIBED IN [9].

```
[H,PdB,PmdB] = noiseLoad(N,Q,L)

% H : Frequency response of the filter.
% PdB : Noise PSD in dB for each test.
% PmdB : Average noise PSD in dB.
% N : Network matrix.
% Q : Quantizer matrix.
% L : Number of tests performed.

m = length(find(N(:,1)-N(:,2)<0)); % Number of delays in the network
ci = zeros(1,m);

% Initialization

pk = 1024;
sumH = zeros(1,pk);
sumY2 = sumH;

% Test Loop

for i = 1:L
    phi = 2*pi*rand(1,pk/2-1);
    phi = [0 phi 0 -phi(pk/2-1:-1:1)];
    Vp = exp(j*phi);
    vp = real(ifft(Vp));
    v = [vp vp];
    [y,cf] = filterNq(N,v,ci,Q);
    y = y(pk+1:1:2*pk);
    Yp = fft(y);
    sumH = sumH+Yp./Vp;
    sumY2 = sumY2+real(Yp.*conj(Yp));
end

H = sumH/L;
P = ((sumY2/L)-real(H.*conj(H)))/pk;
PdB = 10*log10(P);
PmdB = 10*log10(mean(P));
```

with $b_0 = b_1 = b_2 = 0.18$, $c_0 = 0.5625$, $c_1 = -0.75$ and $c_2 = 1$. Regarding the quantization matrix, and following [9], floating point arithmetic (`mode = 'double'`) is used for all the quantizers except for q_{22}, q_{42}, q_{52}, q_{72} and q_{82}, which quantize the values of the outputs of the three adders in Figure 5 whenever they are used. For these quantizers we use a signed fixed-point quantizer (`mode = 'fixed'`), rounding towards $-\infty$ (`roundmode = 'floor'`), saturating in overflow (`overflowmode = 'saturate'`), using 15 bits for the fractional part and 1 bit for the integer part (i.e., 16 bits overall). Now we apply again the noise load method to estimate the frequency response of the filter, altogether with its noise PSD, under the presence of FWL effects [9]. By performing 100 iterations of the method, we obtain the estimates of the frequency response of the filter, $\hat{H}(\omega)$, and the noise power spectral density, $\hat{N}(\omega)$, shown in Figure 6. Note that these results are identical to the ones displayed in Figure 4 of [9], showing that we are able to obtain the same resolution, but following a much more reproducible approach.

V. CONCLUSIONS

In this paper, we have introduced a novel compact framework for the representation of single-input single-output (SISO) linear digital networks. This framework allows us to describe precisely the data flow through the network as well as all the quantizations performed, thus allowing for an easily reproducible analysis of any of the networks described. As case studies for the proposed approach, we have analyzed a third order Butterworth filter with four different implementations, as well as the noise power spectral density of a second-order

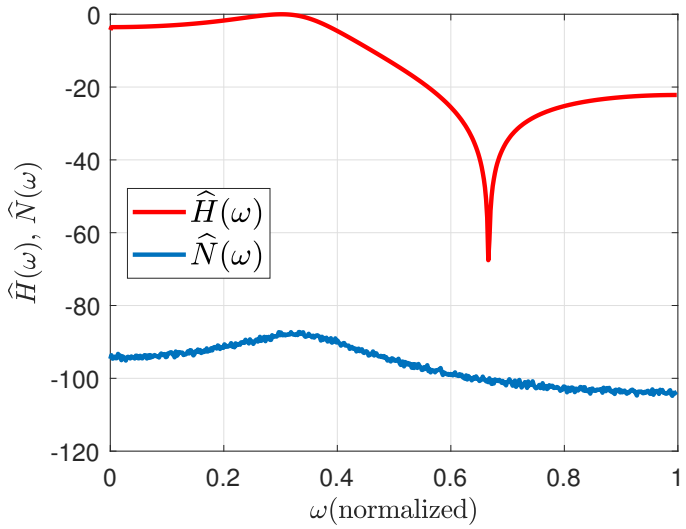

Fig. 6. Estimated frequency response of the filter, $\hat{H}(\omega)$, and noise PSD, $\hat{N}(\omega)$, for the example in Section IV-B.

system. Future lines include analyzing more complex systems, extending the current framework to multiple-input multiple-output (MIMO) networks, fully exploiting the multiple word length paradigm that can be naturally integrated within the current framework, and combining it with the Monte Carlo analysis introduced in [10] to optimize the number of quantization bits used for each coefficient and operation.

ACKNOWLEDGMENT

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness through projects TEC2015-64835-C3-1-R and TEC2015-64835-C3-3-R.

REFERENCES

[1] P. Vandewalle, J. Kovacevic, and M. Vetterli, "Reproducible research in signal processing," *IEEE Signal Processing Magazine*, vol. 26, no. 3, pp. 37–47, 2009.

[2] D. L. Donoho, "An invitation to reproducible computational research," *Biostatistics*, vol. 11, no. 3, pp. 385–388, 2010.

[3] J. Freire, P. Bonnet, and D. Shasha, "Computational reproducibility: state-of-the-art, challenges, and database research opportunities," in *Proceedings of the 2012 ACM SIGMOD international conference on management of data*. ACM, 2012, pp. 593–596.

[4] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, 4th ed. McGraw-Hill, 2010.

[5] D. Osés, F. Cruz-Roldán, M. Blanco-Velasco, and S. L. Netto, "A single matrix representation for general digital filter structures," *IEEE Signal Processing Mag.*, vol. 28, no. 6, pp. 143–148, Nov. 2011.

[6] P. Diniz, E. da Silva, and S. Netto, *Digital Signal Processing: System Analysis and Design*, 2nd ed. Cambridge University Press, 2010.

[7] R. E. Crochiere and A. V. Oppenheim, "Analysis of linear digital networks," *Proc. of the IEEE*, vol. 63, no. 4, pp. 581–595, Apr. 1975.

[8] T. Hilaire, P. Chevrel, and J. F. Whidborne, "A unifying framework for finite wordlength realizations," *IEEE Transactions on Circuits and Systems I*, vol. 54, no. 8, pp. 1765–1774, Aug. 2007.

[9] H. Schussler and Y. Dong, "A new method for measuring the performance of weakly nonlinear systems," in *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 1989, pp. 2089–2092.

[10] D. Luengo, D. Osés, and L. Martino, "Monte Carlo limit cycle characterization," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Florence (Italy), 4–9 May 2014, pp. 8043–8047.