FIXED-POINT IMPLEMENTATION OF LATTICE WAVE DIGITAL FILTER: COMPARISON AND ERROR ANALYSIS

Anastasia Volkova, Thibault Hilaire

LIP6, Pierre and Marie Curie University (UPMC Univ Paris 06), Paris, France

ABSTRACT

A consistent analysis of design filter and its further implementation in fixed-point arithmetic requires a large amount of work, and this process differs from one filter representation to another. For the unifying purposes of such flow a Specialized Implicit Form (SIF) had been proposed in [1].Various sensitivity and stability measures have been adapted to it along with an *a priori* error analysis (quantification of the coefficients and output error). In this paper a conversion algorithm for the widely used Lattice Wave Digital Filters (LWDF) to the SIF is presented, along with finite precision error analysis. It allows to compare fairly LWDF to other structures, like direct forms and state-space, as illustrated with the numerical example.

***Index Terms*—** Filter implementation, Lattice Wave Digital Filters, error analysis, fixed-point arithmetic.

1. INTRODUCTION

Most of control and signal processing algorithms are implemented for application in embedded systems, which use finite-precision arithmetic. Unfortunately, the quantization of the coefficients and the roundoff errors in the computations lead to degradation of the algorithms. This makes the implementation process tedious and error-prone, since no automatic tool exists for the filter-to-code transformation.

Moreover, this process should be able to take into consideration the diversity of Infinite Impulse Response (IIR) filter realizations (Direct Forms, state-space, Wave, etc.). These realizations are equivalent mathematically but not anymore in finite-precision arithmetic. Their structure make some of them more *sensitive* to the quantization of the coefficients, whereas some of them are very *robust* to roundoff errors.

Furthermore, each representation demands its own analysis procedure. For these reasons, a unifying framework (denoted SIF) was introduced in [1]. It allows to represent any filter in a unified form and then to apply a rigorous error analysis and some optimization methods, such as transfer function and pole sensibility measures or output error. Moreover, automatic Fixed-Point code generation (with optimization in term of implementation cost) was proposed in [2]. The complete flow of filter implementation using SIF is shown in Figure 1. Moreover, since this process is unified, other realizations of the same filter may be simultaneously considered and a fair comparison can be provided.

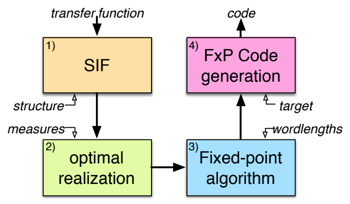


Fig. 1: From transfer function to Fixed-Point code.

In this article, we consider Lattice Wave Digital Filters (LWDF) as a target filter realization which we convert to SIF and show several filter analysis measure results in comparison with Direct form I (DFI), ρ Direct Form II transposed (ρDFIIt [3]) and state-space realizations. For that purpose, Lattice Wave Digital Filters are reminded in Section 2. A fixed-point arithmetic error analysis for the SIF is detailed in section 3 and a LWDF-to-SIF conversion algorithm is exhibited in Section 4. Finally, an illustrative example is given in order to compare various realizations.

2. LATTICE WAVE DIGITAL FILTERS

Lattice Wave Digital Filters is a class of recursive Wave Digital Filters that inherit several good properties, such as stability for implementation and possibility of suppression of parasitic oscillations. LWDFs can be either derived from an analog reference filter [4] or using explicit formulas [5].

The LWDF structure is highly modular and has a high degree of parallelism, which makes them suitable for a VLSI implementation. Their good stability qualities [4] make them good candidates for adaptive filtering [6] and Hilbert transformers design [7].

LWDF is represented by two parallel branches, which realize all-pass filters. These all-pass filters are composed of cascaded first- and second-order symmetric two-port adaptors. Its data-flow diagram (DFG) is shown in Figure 2.

Each adaptor contains three adders and one multiplier. According to [5], the adaptor coefficients γ may be guaranteed to fall into the interval $]-1,1[$. In [4] it was proposed to use Richards' structures for adaptors, as on Figure 3.

This work has been sponsored by french ANR agency under grant No ANR-13-INSE-0007.

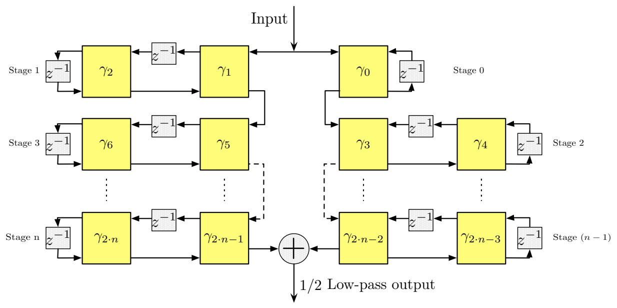


Fig. 2: Data-flow diagram of a LWD Filter.

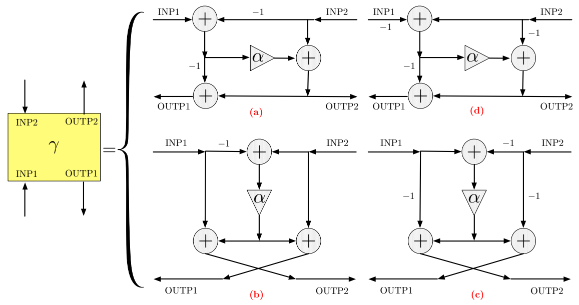


Fig. 3: Two-port adaptor structures, for which actual multiplier α is computed out of γ using Table 1.

Moreover, instead of multiplication by γ an easy-to-implement multiplication by $\alpha \in]0, 1/2]$ takes place in each type of structure, as summarized in the Table 1.

Type	γ range	Value of α
a	$1/2 < \gamma < 1$	$\alpha = 1 - \gamma$
b	$0 < \gamma \leqslant 1/2$	$\alpha = \gamma$
c	$-1/2 \leqslant \gamma < 0$	$\alpha = \|\gamma\|$
d	$-1 < \gamma < -1/2$	$\alpha = 1 + \gamma$

Table 1: γ to α conversion for different γ ranges.

The transfer function of a LWDF and its complementary transfer function are power complementary. This means, that the high-pass filter may be obtained by changing the sign of all-pass lower filter. Therefore second output may be obtained simultaneously at no cost. Band-pass and pass-band filters are obtained by cascading low- and high-pass filters.

In [5] explicit formulas for LWDF transfer function design for several common filters such as Butterworth, Cauer (elliptic) and Chebyshev were presented. However, LWDF structure can realize all transfer functions that can be achieved by the families of reference filters considered so far, although not strictly all. Due to its good qualities, LWDFs are considered in numerous different applications, including studies on linear-phase structuresand energy-efficient structures [8]. However, all studies on lattice wave structures implementation in finite word-length arithmetic are performed **a posteriori**, *i.e.* when the implementation parameters are known [9]. These are specific to the LWDF and are not really suitable for a fair comparison with other structures.

In this work, however, we apply a more arithmetic approach on LWDF using SIF, which provides **a priori** bounds on the coefficient's quantization errors and on roundoff errors of computations.

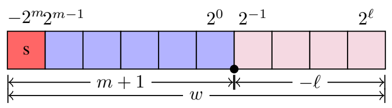


Fig. 4: Fixed-point representation (here, $m = 5$ and $\ell = -4$).

3. FIXED-POINT ARITHMETIC ERROR ANALYSIS

3.1. Specialized Implicit Framework

In order to encompass all the possible realizations for a given transfer function the Specialized Implicit Framework has been proposed in [1]. All the input-output relationships with delays, computation order, multiplications by constants and additions can be represented with the SIF. This macroscopic description is then more suited for the analysis than a graph relationship as it gives direct analytical formula for the finite precision error analysis [1]. It is presented here for Single Input Single Output (SISO) filters, but it can be easily extended to Multiple-Inputs Multiple Outputs (MIMO) ones.

It is an extension of the state-space framework, modified in order to allow chained Sum-of-Products (SoP) operations. Variables that are stored from one step to the other are regrouped in the state vector $\boldsymbol{x}(k)$, while intermediate results are collected in the $\boldsymbol{t}(k)$ vector. $u(k)$ and $\boldsymbol{y}(k)$ represent the input and output, respectively:

$$\left\{\begin{aligned} \boldsymbol{J}\boldsymbol{t}(k+1) &= & \boldsymbol{M}\boldsymbol{x}(k) &+ \boldsymbol{N}u(k) \\ \boldsymbol{x}(k+1) &= \boldsymbol{K}\boldsymbol{t}(k+1) &+ \boldsymbol{P}\boldsymbol{x}(k) &+ \boldsymbol{Q}u(k) \\ y(k) &= \boldsymbol{L}\boldsymbol{t}(k+1) &+ \boldsymbol{R}\boldsymbol{x}(k) &+ \boldsymbol{S}u(k) \end{aligned}\right. \tag{1}$$

Note that $\boldsymbol{J}$ is lower triangular with 1 on its diagonal, so the first value of the vector $\boldsymbol{t}(k+1)$ is first computed, and then the second one is computed using the first and so on (thus,the computation of $\boldsymbol{J}^{-1}$ is not necessary)). The *implicit* term $\boldsymbol{J}\boldsymbol{t}(k+1)$ naturally serves for preservation of the order of computations specific for each realization.

The matrix $\boldsymbol{Z}$ denotes the set of the coefficients

$$\boldsymbol{Z} \triangleq \begin{pmatrix} -\boldsymbol{J} & \boldsymbol{M} & \boldsymbol{N} \\ \boldsymbol{K} & \boldsymbol{P} & \boldsymbol{Q} \\ \boldsymbol{L} & \boldsymbol{R} & \boldsymbol{S} \end{pmatrix}. \tag{2}$$

Various analytical measures [1, 10, 11] have been introduced for this framework, along with rigorous error analysis algorithms and optimal (in respect to several different criteria, such as error bound on roundoff errors) fixed-point codegeneration tool [2].

Section 4 presents how to express the LWDF structure in this formalism, in order to apply SIF's analytical capabilities.

3.2. Fixed-Point Arithmetic

In two's complement Fixed-Point (FxP) arithmetic, a FxP number x is represented by

$$x = -2^m x_m + \sum_{i=\ell}^{m-1} 2^i x_i, \qquad (3)$$

where x_i is the i^{th} bit of x, and m and ℓ are the *Most Significant Bit* (MSB) and *Least Significant Bit* (LSB) of x (see Figure 4). The tuple (m, ℓ) defines the Fixed-Point format of such a FxP number.

If a real non null constant c has to be approximated by a w-bit FxP number, then its MSB is first deduced from

$$m = \lfloor \log_2 |c| \rfloor + 1 \qquad (4)$$

and then its LSB ℓ is deduced with $\ell = m - w + 1$ (eq. (4) may be wrong in some very special cases, see [12] for the complete algorithm). c is approximated with an absolute error Δc such that $|\Delta c| \leqslant 2^{\ell-1}$ if round-to-nearest mode is chosen.

If x is a variable known to be in the interval $[\underline{x}; \overline{x}]$, then the appropriate w-bit FxP format is deduced from the FxP format of $\underline{x}$ and $\overline{x}$, and those with largest MSB is chosen for x.

3.3. Coefficient's quantization

Obviously, after implementation the coefficients $\boldsymbol{Z}$ will be modified into $\boldsymbol{Z} + \boldsymbol{\Delta Z}$, and the errors $\boldsymbol{\Delta Z}$ depend on their LSB. In order to evaluate how the transfer function may be modified by the quantization of the coefficients, sensitivity-based measures are usually used [13].

The sensitivity of the transfer function H with respect to the coefficients $\boldsymbol{Z}$ is given by $\frac{\partial H}{\partial \boldsymbol{Z}}$. This term is a MIMO transfer function that can be expressed analytically in state-space form [1]. Unfortunately, this commonly used measure is not fair and does not reflect how the coefficients' quantization changed the transfer function H into $H + \Delta H$ since the absolute error of the coefficients may not all have the same magnitude order. For that purpose, a *stochastic* sensitivity-based measure has been proposed and developed with respect to real FxP considerations in [10].

Here, quantization error $\boldsymbol{\Delta Z}_{ij}$ of the coefficient $\boldsymbol{Z}_{ij}$ is considered as a random variable, uniformly distributed in $[-2^{\ell_{\boldsymbol{Z}_{ij}}-1}; 2^{\ell_{\boldsymbol{Z}_{ij}}-1}]$ where $\ell_{\boldsymbol{Z}_{ij}}$ is the LSB of the coefficient $\boldsymbol{Z}_{ij}$ (as introduced in section 3.2). Then, the error transfer function ΔH can be seen as a transfer function with random variables as coefficients. Its second-order moment is defined as

$$\sigma^2_{\Delta H} \triangleq \frac{1}{2\pi} \int_0^{2\pi} E\left\{ \left| \Delta H \left(e^{j\omega} \right) \right|^2 \right\} d\omega, \qquad (5)$$

where $E\{.\}$ is the expectation operator. It reflects how much the transfer function H has changed due to the quantization. From [10], it can be evaluated by

$$\sigma^2_{\Delta H} = \left\| \frac{\partial H}{\partial \boldsymbol{Z}} \times \boldsymbol{\Xi} \right\|_2^2 \qquad (6)$$

where

$$\Xi_{ij} \triangleq \begin{cases} 0 & \text{if } \boldsymbol{Z}_{ij} \in \{0, \pm 1\} \\ \frac{2^{-w_{\boldsymbol{Z}_{ij}}+1}}{\sqrt{3}} \lfloor \boldsymbol{Z}_{ij} \rfloor_2 & \text{otherwise} \end{cases} \qquad (7)$$

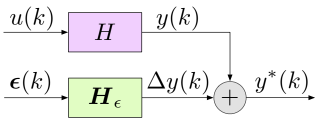


Fig. 5: The implemented filter can be seen as the exact filter perturbed by the roundoff errors.

and $\times$ is the entrywise product, $\lfloor x \rfloor_2$ is the nearest power of 2 lower than x and $w_{\boldsymbol{Z}_{ij}}$ the word-length used to represent $\boldsymbol{Z}_{ij}$. This permits $\sigma^2_{\Delta H}$ to capture more information on transfer function error.

If all the coefficients have the same word-length, then a normalized transfer function error measure [10] is defined by

$$\bar{\sigma}^2_{\Delta H} \triangleq \left\| \frac{\partial H}{\partial \boldsymbol{Z}} \times \lfloor \boldsymbol{Z} \rfloor_2 \right\|_2^2. \qquad (8)$$

This normalization is useful for a concrete realization analysis on the design stage, when word-lengths are not known.

The same approach was applied for the poles $\{\boldsymbol{\lambda}_i\}$ of the systems, or more interestingly to their moduli $|\boldsymbol{\lambda}_i|$ in order to ensure filter's stability after quantization. The poles moduli sensitivity (w.r.t. the coefficients) is measured with $\frac{\partial |\boldsymbol{\lambda}_i|}{\partial \boldsymbol{Z}}$ and detailed in [13]. For the same reason as the transfer function sensitivity, it was extended as the second-order moment of the random variables $\Delta |\boldsymbol{\lambda}_i|$ and analogous to (6) pole error $\sigma^2_{\Delta|\boldsymbol{\lambda}|}$ was introduced. Likewise to (8) the word-length independent normalized error pole measure $\bar{\sigma}^2_{\Delta|\boldsymbol{\lambda}|}$ may be computed.

3.4. Roundoff errors

In addition to the quantization of the coefficients, the second source of finite precision degradation of the exact filter is the *roundoff errors* that occur during the computations. Due to binary-point alignments and the result quantizations, the sum-of-products (SoPs) operations aimed to compute the output from the input are not exact and produce errors.

If some extra guard bits added, the SoPs can be performed with faithful rounding of the las bit, *i.e.* with an error ε such that $|\varepsilon| \leqslant 2^\ell$ where ℓ is the LSB of the result [11].

Considering these errors in implemented algorithm (1) leads to the following implemented filter:

$$\begin{aligned} \boldsymbol{J}\boldsymbol{t}^*(k+1) &= \qquad\qquad\qquad \boldsymbol{M}\boldsymbol{x}^*(k) + \boldsymbol{N}u(k) + \boldsymbol{\varepsilon_t}(k) \\ \boldsymbol{x}^*(k+1) &= \boldsymbol{K}\boldsymbol{t}^*(k+1) + \boldsymbol{P}\boldsymbol{x}^*(k) + \boldsymbol{Q}u(k) + \boldsymbol{\varepsilon_x}(k) \\ y^*(k) &= \boldsymbol{L}\boldsymbol{t}^*(k+1) + \boldsymbol{R}\boldsymbol{x}^*(k) + \boldsymbol{S}u(k) + \varepsilon_y(k) \end{aligned} \qquad (9)$$

where $\boldsymbol{\varepsilon}(k) \triangleq \left(\boldsymbol{\varepsilon_t}^\top(k), \boldsymbol{\varepsilon_x}^\top(k), \varepsilon_y^\top(k) \right)^\top$ is the vector of all the roundoff errors due to the FxP computations of the SoPs. Performing the difference between an exact (1) and implemented filter (9), the output error $\Delta y(k) \triangleq y^*(k) - y(k)$ can be seen as the output of the roundoff error vector $\boldsymbol{\varepsilon}(k)$ through a (MIMO) error filter denoted $\boldsymbol{H}_\varepsilon$, as shown in Figure 5. $\boldsymbol{H}_\varepsilon$ can be easily described as a state-space [12].

The worst possible output error $\|\Delta y\|_\infty \triangleq \sup_{k\geqslant 0} |\Delta y(k)|$ can be deduced from a bound $\|\boldsymbol{\varepsilon}\|_\infty$ on the error vector $\boldsymbol{\varepsilon}(k)$:

$$\|\Delta y\|_\infty = \langle\!\langle \boldsymbol{H}_\varepsilon \rangle\!\rangle \, \|\boldsymbol{\varepsilon}\|_\infty \tag{10}$$

where $\langle\!\langle \boldsymbol{H}_\varepsilon \rangle\!\rangle$ is the Worst Case Peak Gain matrix of the MIMO filter $\boldsymbol{H}_\varepsilon$, *i.e.* the L_1-norm of its impulse response. This matrix can be evaluated at any arbitrary precision [14].

The error vector is bounded by $\|\boldsymbol{\varepsilon}\|_\infty \leqslant 2^{\boldsymbol{\ell_{txy}}}$, where $\boldsymbol{\ell_{txy}} = \boldsymbol{m_{txy}} - \boldsymbol{w_{txy}} + 1$ is the vector of the LSB of the variables $\boldsymbol{t}$, $\boldsymbol{x}$ and y. These LSB are determined according to the word-lengths $\boldsymbol{w_{txy}}$ and their respective MSB $\boldsymbol{m_{txy}}$.

The word-lengths $\boldsymbol{w_{txy}}$ are an implementation choice (determined by the hardware architecture used for the implementation), whereas the MSBs $\boldsymbol{m_{txy}}$ are expressed as logarithm of the largest possible values for the variables $\boldsymbol{t}$, $\boldsymbol{x}$ and y:

$$\boldsymbol{m_{txy}} = \left\lfloor \log_2 \left(\langle\!\langle \boldsymbol{H}_u \rangle\!\rangle \, \|u\|_\infty \right) \right\rfloor + 1 \tag{11}$$

where $\|u\|_\infty$ is the largest value of the input and $\boldsymbol{H}_u$ is the transfer function from the input to the variables $\boldsymbol{t}$, $\boldsymbol{x}$ and y. $\boldsymbol{H}_u$ is a MIMO transfer function that can be easily described as the state-space [12].

Moreover, if the largest possible value of the input is a power-of-2, then substituting (11) into (10) leads to a normalized output error bound (independent of word-lengths analogously to Section 3.3):

$$\overline{\Delta y} \triangleq \langle\!\langle \boldsymbol{H}_\varepsilon \rangle\!\rangle \, 2^{\left\lfloor \log_2 \langle\!\langle \boldsymbol{H}_u \rangle\!\rangle \right\rfloor}. \tag{12}$$

4. LWD FILTER TO SIF CONVERSION ALGORITHM

Due to the high modularity of the LWDF structure, the conversion from its DFG to SIF is not difficult. As seen on Figure 2, LWDF consists of two branches, and each branch is a cascade of stages. Each stage in its turn may be considered as a cascade of *subsystems* of two types, as shown on Figure 6.

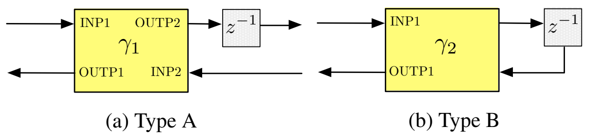


Fig. 6: A stage is considered as cascade of subsystems of type A (a) or type B (b).

Therefore, the basic brick for cascade sequence is actually not a simple adaptor but a two-port adaptor with one output delayed (Type A: Figure 6a) and a 1-input/1-output adaptor with delay (Type B: Figure 6b). Then, given filter's coefficients γ, the conversion algorithm can be divided into following steps:

1. Deduce SIF representation (matrix $\boldsymbol{Z}$) for each subsystem according to its γ value;
2. Cascade subsystems into stages;
3. Cascade stages into branches;
4. Regroup SIFs of the resulting branches into final filter.

The first step can be easily done by applying SIF notation to their DFG. Two subsystems for each adaptor structure must be considered, overall eight basic bricks. For example, for $\gamma \in]-1, 1/2[$ structures to be converted into SIF can be described with following DFGs:

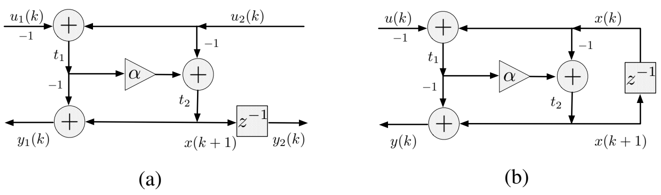


Fig. 7: (a) Type A subsystem with adaptor of type d. (b) Type B subsystem with adaptor of type d.

Applying definition (1), one can deduce that SIFs $\boldsymbol{Z}_A$ and $\boldsymbol{Z}_B$ for subsystems shown on Figure 7 are:

$$\boldsymbol{Z}_A \triangleq \left(\begin{array}{c:c:c} -\boldsymbol{J}_A & \boldsymbol{M}_A & \boldsymbol{N}_A \\ \hdashline \boldsymbol{K}_A & \boldsymbol{P}_A & \boldsymbol{Q}_A \\ \hdashline \boldsymbol{L}_A & \boldsymbol{R}_A & \boldsymbol{S}_A \end{array}\right) = \left(\begin{array}{cc:c:cc} -1 & 0 & 0 & 1 & 1 \\ \alpha & -1 & 0 & 0 & 1 \\ \hdashline 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ \hdashline 0 & 0 & 1 & 0 & 0 \end{array}\right) \tag{13}$$

$$\boldsymbol{Z}_B \triangleq \left(\begin{array}{c:c:c} -\boldsymbol{J}_B & \boldsymbol{M}_B & \boldsymbol{N}_B \\ \hdashline \boldsymbol{K}_B & \boldsymbol{P}_B & \boldsymbol{Q}_B \\ \hdashline \boldsymbol{L}_B & \boldsymbol{R}_B & \boldsymbol{S}_B \end{array}\right) = \left(\begin{array}{cc:c:c} -1 & 0 & 1 & -1 \\ \alpha & -1 & 1 & 0 \\ \hdashline 0 & -1 & 0 & 0 \\ \hdashline -1 & 1 & 0 & 0 \end{array}\right) \tag{14}$$

Sequential cascading of two SIFs can also be expressed. For example, if two SIFs are determined with matrices $\{\boldsymbol{J}_1, \boldsymbol{K}_1, \ldots, \boldsymbol{S}_1\}$ and $\{\boldsymbol{J}_2, \boldsymbol{K}_2, \ldots, \boldsymbol{S}_2\}$ respectively, then the cascaded SIF may be explicitly obtained via:

$$\boldsymbol{Z} = \left(\begin{array}{ccc:cc:c} -\boldsymbol{J}_1 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{M}_1 & \boldsymbol{0} & \boldsymbol{N}_1 \\ \boldsymbol{L}_1 & -\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{R}_1 & \boldsymbol{0} & \boldsymbol{S}_1 \\ \boldsymbol{0} & \boldsymbol{N}_2 & -\boldsymbol{J}_2 & \boldsymbol{0} & \boldsymbol{M}_2 & \boldsymbol{0} \\ \hdashline \boldsymbol{K}_1 & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{P}_1 & \boldsymbol{0} & \boldsymbol{Q}_1 \\ \boldsymbol{0} & \boldsymbol{Q}_2 & \boldsymbol{K}_2 & \boldsymbol{0} & \boldsymbol{P}_2 & \boldsymbol{0} \\ \hdashline \boldsymbol{0} & \boldsymbol{S}_2 & \boldsymbol{L}_2 & \boldsymbol{0} & \boldsymbol{R}_2 & \boldsymbol{0} \end{array}\right). \tag{15}$$

Notate, that even on this first step of cascading the matrices Z_A and Z_B are sparse. Applying cascade formula (15) would produce an even more sparse matrix with just two occurrences of coefficients α.
An example of matrice $\boldsymbol{Z}$ is exhibited in the next section.

5. NUMERICAL EXAMPLES AND COMPARISONS

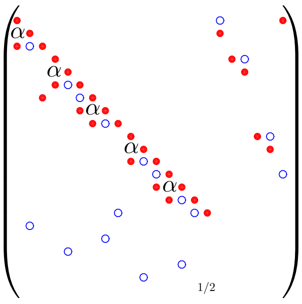
Following example is based on LWDF coefficients, which were obtained with Wave Digital Filters[1] toolbox for Matlab. This toolbox is based on explicit formulas introduced in [5]. However, the LWDF to SIF conversion algorithm requires only the γs, therefore any procedure from [15] can also

[1]`http://ens.ewi.tudelft.nl/~huib/mtbx/index.php`

be used as preliminary step. The FWR toolbox[2] for Matlab was used for the SIF analysis.

A Matlab generated low-pass 5^{th} Butterworth filter with cutoff frequency 0.1 was considered. Four realizations of this filter were considered: LWDF, balanced state-space, Direct Form I and ρ Direct Form II transposed. Only the normalized versions of measures introduced in Section 3 were computed. Result SIF form for LWDF structure is a sparse matrix of size 22×22 shown below, where alpha are adaptor coefficients, filled circles represent 1 and contour circles -1:

$\mathbf{Z} =$ (16)



As described in [16, 3], the ρDFIIt is parametrized by n extra parameters. It can be a subject of multi-criteria optimization. For this example a tradeoff function minimizing weighted sum of normalized transfer function and pole errors was used. Excellent results may be observed in Table 2.
However, Direct Form I showed to be very ill-conditioned for the considered filter, and its pole error cannot be computed. The state-space realization chosen is a balanced one. It is expectantly good at output error, but more sensitive to quantization (it has more coefficients).

Thanks to the minimal number of coefficients, the LWDF approaches in its normalized transfer function error the optimized ρDFIIt. The specific alternating distribution of LWDF poles [9] leads to good results in pole error measure. Normalized output error is, however, quite large because it was not designed to minimize the propagation of roundoff errors. Therefore, ρDFIIt may be a good alternative, but at a cost of 6 more coefficients. To provide a consistent implementation comparison a code optimization and generation chain FiPoGen [11] (plus FloPoCo[3] for FPGA) is required.

6. CONCLUSION

Various studies on LWDF have been introduced over the years. However, existing computational error analysis are dedicated to that particular structure and do not really permit comparisons. Conversion of LWDF to SIF permits to apply numerous classical and novel sensibility measures upon any LWDF realization and any other.
Further work will consist of using the Fixed-Point Generator [11] capabilities to produce optimal (with respect to either implementation cost and some error criteria) FxP code for various hardware and software architectures and complete the filter-to-code transformation.

[2]`https://gforge.inria.fr/projects/fwrtoolbox/`
[3]`http://flopoco.gforge.inria.fr/`

Realization	size(Z)	coeff.	$\bar{\sigma}^2_{\Delta H}$	$\bar{\sigma}^2_{\Delta\|\lambda\|}$	$\overline{\Delta_y}$
LWDF	22×22	5	0. 3151	0.56	122.9
state-space	6× 6	36	1.15	5.75	23.33
ρDFIIt	11×11	11	0.09	0.45	94.3
DFI	12×12	11	1.42e+6	-	7.961

Table 2: Different realizations comparison.

REFERENCES

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

[2] B. Lopez, *Implémentation optimale de filtres linéaires en arithmétique virgule fixe*, Ph.D. thesis, UPMC, 2015.

[3] Z. Zhao and G. Li, "Roundoff noise analysis of two efficient digital filter structures," *IEEE Transactions on Signal Processing*, vol. 54, no. 2, pp. 790–795, February 2006.

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

[5] L. Gazsi, "Explicit formulas for lattice wave digital filters," *IEEE Trans. Circuits & Systems*, vol. 32, no. 1, 1985.

[6] B. Friedlander, "Lattice filters for adaptive processing," *Proceedings of the IEEE*, vol. 70, no. 8, pp. 829–867, Aug 1982.

[7] H. Johansson and L. Wanharomar, "Digital hilbert transformers composed of identical allpass subfilters," in *ISCAS 1998. Proceedings of*, vol. 5, pp. 437–440 vol.5.

[8] H. Ohlsson O. Gustafsson W. Li L. Wanhammar, "An environment for design and implementation of energy efficient digital filters," in *SSoCC*, apr 2003.

[9] J. Yli-Kaakinen and T. Saramäki, "A systematic algorithm for the design of lattice wave digital filters with short-coefficient wordlength," *IEEE Trans. on Circuits & Systems*, 2007.

[10] T. Hilaire and P. Chevrel, "Sensitivity-based pole and input-output errors of linear filters as indicators of the implementation deterioration in fixed-point context," *EURASIP Journal on Advances in Signal Processing*, January 2011.

[11] B. Lopez, T. Hilaire, and L.-S. Didier, "Formatting bits to better implement signal processing algorithms," in *4th Int. conf. PECCS, proceedings of*, 2014.

[12] T. Hilaire and B. Lopez, "Reliable implementation of linear filters with fixed-point arithmetic," in *IEEE Workshop on Signal Processing Systems, SiPS 2013*, 2013, pp. 401–406.

[13] M. Gevers and G. Li, *Parametrizations in Control, Estimation and Filtering Probems*, Springer-Verlag, 1993.

[14] A. Volkova, T. Hilaire, and C. Lauter, "Reliable evaluation of the Worst-Case Peak Gain matrix in multiple precision," in *IEEE Symposium on Computer Arithmetic*, 2015.

[15] A. Fettweis H. Levin and A. Sedlmeyer, "Wave digital lattice filters," *Int. J. Circ. Theor. Appl.*, vol. 2, pp. 203–211, 1974.

[16] G. Li, C. Wan, and G. Bi, "An improved rho-DFIIt structure for digital filters with minimum roundoff noise," *IEEE Trans. on Circuits & Systems*, vol. 52, no. 4, pp. 199–203, 2005.