Bit Accurate Roundoff Noise Analysis of Fixed-Point Linear Controllers

Thibault HILAIRE, Daniel MÉNARD and Olivier SENTIEYS
IRISA/IRISA - Cairn project, University of Rennes 1, FRANCE
surname.name@irisa.fr

***Abstract*— The analytic evaluation of roundoff noise is an interesting approach to analyze the effects of fixed-point implementation of linear filters or controllers. This paper is based on a generic framework that describes controller algorithms. Fixed-point implementation (in different schemes) that is associated to these algorithms are exhibited, and the output roundoff noise power is then analytically build. Finally, the optimal realization problem, according to the roundoff noise power, is considered.**

I. Introduction

The majority of signal processing and control systems are digitally implemented into general purpose microprocessors, DSP[1] or specific computing devices (ASIC[2], FPGA[3], etc.). Since the processor cannot compute with infinite precision, the fixed-point implementation of filters/controllers leads to deterioration in performance. They have two separate origins [3] corresponding to the quantization of embedded coefficients and the roundoff errors in the numerical computations. They can respectively be formalized as parametric errors and numerical noises. If the controller and the target architecture are specified, they both depend on the realization (the mathematical algorithm used and its parameters), the fixed-point representation of the variables and the coefficients, and the software/hardware design. Here only roundoff errors (see [5] for parametric errors) are considered.

The roundoff errors have been studied in two different ways: works in [7], [14], [3] deal with a roundoff noise measure (the Roundoff Noise Gain) unlinked to hardware considerations in order to optimize the realization with respect to that criteria, whereas works in [12], [10] deal with the Signal Quantization Noise Ratio and are more focused on the software/hardware realizations.
Since the equivalent realizations and the HW/SW considerations are not yet exploited together, this work aims at combining these two approaches and provides a new roundoff noise power expression linked to two precise fixed-point implementation schemes. It is a part of a more global approach, from the filter/controller design to the optimization of the fixed-point coefficients and the final code generation. A Toolbox for Matlab (*FWR Toolbox*[4]) has been specially developed for this approach.

The objective of this paper is to exhibit the precise fixed-point specification of any given controller realization and to evaluate the global roundoff noise power. Then, since various algorithms exists to numerically realize a linear time invariant filter, it is of interest to consider the *optimal* implementation problem.

This paper is organized as follows. After presenting the implicit state-space framework in section II, a general roundoff noise analysis, applied to this form, is exhibited in section III. The automatic fixed-point implementation is presented in section IV and roundoff noise power in section V. The optimal design problem and some numerical results are finally proposed in section VI and VII.

II. The Implicit State-Space Framework

Work in [5] highlights the interest of the implicit state-space representation in the context of Finite Word Length (FWL) implementation problems and proposes to use a specialized form directly connected to the in-line computations to be performed. It can be used as a unifying framework to allow a more detailled (but macroscopic) description of FWL implementations. Various realizations, like q (shift) or δ-realizations, classical Direct Forms I and II, cascade or parallel decompositions, mixed structures, ... may be then described in a single unifying form [5], [4].

Definition 1 (Implicit specialized form) *Equation* (1) *recalls the specialized implicit form that explicitly expresses the parametrization and the intermediate variables used:*

$$\begin{pmatrix} J & 0 & 0 \\ -K & I_n & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T(k) \\ X(k) \\ U(k) \end{pmatrix} \quad (1)$$

where

- $J \in \mathbb{R}^{l\times l}$, $K \in \mathbb{R}^{n\times l}$, $L \in \mathbb{R}^{p\times l}$, $M \in \mathbb{R}^{l\times n}$, $N \in \mathbb{R}^{l\times m}$, $P \in \mathbb{R}^{n\times n}$, $Q \in \mathbb{R}^{n\times m}$, $R \in \mathbb{R}^{p\times n}$, $S \in \mathbb{R}^{p\times m}$, $T(k) \in \mathbb{R}^l$, $X(k) \in \mathbb{R}^n$, $U(k) \in \mathbb{R}^m$ *and* $Y(k) \in \mathbb{R}^p$,
- $U(k)$ *represents the* m *inputs, and* $Y(k)$ *the* p *outputs,*
- $X(k+1)$ *is the* n *stored states* ($X(k)$ *is effectively stored from one step to the next, in order to compute* $X(k+1)$ *at step* k*),*
- $T(k+1)$ *is the* l *intermediate variables in the calculations of step* k *(the column of* 0 *in the second matrix shows that* $T(k)$ *is not used for the calculation at step* k*: that characterizes the concept of intermediate variables),*
- *the matrix* J *is lower triangular with* 1 *on the diagonal.*

[1] Digital Signal Processors
[2] Application-Specific Integrated Circuit
[3] Field Programmable Gate-Array
[4] Sources available at `http://fwrtoolbox.gforge.inria.fr/`

$T(k+1)$ and $X(k+1)$ form the state-vector: $X(k+1)$ is stored from one step to the next, while $T(k+1)$ is computed and used inside one time step.

It is implicitly considered throughout the paper that the computations associated to the realization (1) are associated to the following algorithm:

(i) intermediate variable computation (J is lower triangular, so the first value of $T(k+1)$, $T_1(k+1)$, is first calculated, and then $T_2(k+1)$ using $T_1(k+1)$ and so on (the computation of J^{-1} is not necessary)):
$J.T(k+1) \leftarrow M.X(k) + N.U(k)$

(ii) state-vector update:
$X(k+1) \leftarrow K.T(k+1) + P.X(k) + Q.U(k)$

(iii) outputs computation:
$Y(k) \leftarrow L.T(k+1) + R.X(k) + S.U(k)$

Steps (ii) and (iii) can be swapped in order to reduce the computational delay from input(s) to output(s).

Equation (1) is equivalent in infinite precision to the classical state-space form:

$$\left(\begin{array}{c} T(k+1) \\ X(k+1) \\ \hline Y(k) \end{array}\right) = \left(\begin{array}{cc|c} 0 & J^{-1}M & J^{-1}N \\ 0 & A_Z & B_Z \\ \hline 0 & C_Z & D_Z \end{array}\right) \left(\begin{array}{c} T(k) \\ X(k) \\ \hline U(k) \end{array}\right) \quad (2)$$

with:

$$\begin{array}{ll} A_Z = KJ^{-1}M + P, & B_Z = KJ^{-1}N + Q, \\ C_Z = LJ^{-1}M + R, & D_Z = LJ^{-1}N + S. \end{array} \quad (3)$$

However, equation (2) corresponds to a different parametrization than the one in eq. (1).

The equivalent transfer function considered is then given by

$$H : z \mapsto C_Z(zI_n - A_Z)^{-1}B_Z + D_Z \quad (4)$$

In the following, a realization $\mathcal{R}$ will be defined in the implicit form by its parameters used for the internal description $\mathcal{R} \triangleq (J, K, L, M, N, P, Q, R, S)$. It could also be equivalently written in a compact form $\mathcal{R} = (Z, l, m, n, p)$ with

$$Z \triangleq \begin{pmatrix} -J & M & N \\ K & P & Q \\ L & R & S \end{pmatrix} \quad (5)$$

The usual realizations (direct forms, state-space, δ-realizations, cascade, parallel, mixed realization, etc.) can be easily expressed in the Implicit State-Space Framework. For example, a δ-state-space realization

$$\left\{ \begin{array}{rcl} \delta[X(k)] & = & A_\delta X(k) + B_\delta U(k) \\ Y(k) & = & C_\delta X(k) + D_\delta U(k) \end{array} \right. \quad (6)$$

where $\delta = \frac{q-1}{\Delta}$ (Δ is strictly positive constant, q is the shift-operator) can be expressed as:

$$\begin{pmatrix} I_n & 0 & 0 \\ -\Delta I_n & I_n & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \begin{pmatrix} 0 & A_\delta & B_\delta \\ 0 & I_n & 0 \\ 0 & C_\delta & D_\delta \end{pmatrix} \begin{pmatrix} T(k) \\ X(k) \\ U(k) \end{pmatrix} \quad (7)$$

This form is well known [3], [13] to be numerically superior to the usual shift-operator, because it generally results in less sensitive implementation with less roundoff noise. Other examples can be found in [5], [4].

III. Roundoff Noise Analysis

A. Preliminaries

The first (μ) and second (σ, ψ) order moments of a noise vector ξ are denoted and defined by:

$$\mu_\xi \triangleq E\{\xi(k)\} \quad (8)$$

$$\psi_\xi \triangleq E\left\{(\xi(k) - \mu_\xi)(\xi(k) - \mu_\xi)^\top\right\} \quad (9)$$

$$\sigma_\xi^2 \triangleq E\left\{(\xi(k) - \mu_\xi)^\top (\xi(k) - \mu_\xi)\right\} = tr(\psi_\xi) \quad (10)$$

where $E\{\cdot\}$ and $tr(\cdot)$ are respectively the *mean* and the *trace* operator.

The following lemma recalls the basic properties of noise transmission through a linear system:

Lemma 1 *Assume the input noise, $U(k)$, to be such that*

$$E\left\{(U(k) - \mu_U)(U(k-l) - \mu_U)^\top\right\} = \delta_{0,l}\psi_U \quad (11)$$

where $\delta_{i,j}$ represents the Kronecker delta. Denote by $Y(k)$ the resulting output of $U(k)$ through the transfer matrix G. If (A, B, C, D) is a state-space realization of G, the first and second order moments of Y are given by:

$$\mu_Y = (C(I - A)^{-1}B + D)\mu_U \quad (12)$$

$$\sigma_Y^2 = tr\left(\psi_U(D^\top D + B^\top W_o B)\right) \quad (13)$$

where W_o is the observability Gramian of G. W_o is the unique solution of the discrete Lyapunov equation:

$$W_o = A^\top W_o A + C^\top C \quad (14)$$

Proof: It is well known that $C(I - A)^{-1}B + D$ is the DC gain and $\sigma_Y^2 = \|G\varphi_U\|_{l_2}^2$, with φ_U such that $\psi_U = \varphi_U\varphi_U^\top$. The l_2-norm expression with grammians is then applied. See [15], [6]. ■

B. Roundoff Noise Analysis

Let us consider a realization $\mathcal{R}$ described with the implicit form(1), with transfer function H. When implemented, the steps (i) to (iii) are modified by the add of noises $\xi_T(k)$, $\xi_X(k)$ and $\xi_Y(k)$:

$$\begin{array}{r} J.T(k+1) \leftarrow M.X(k) + N.U(k) + \xi_T(k) \\ X(k+1) \leftarrow K.T(k+1) + P.X(k) + Q.U(k) + \xi_X(k) \\ Y(k) \leftarrow L.T(k+1) + R.X(k) + S.U(k) + \xi_Y(k) \end{array} \quad (15)$$

These noises added depend on:

- the way the computations are organized (the order of the sums) and done,
- the fixed-point representation of the inputs, the outputs,
- and the fixed-point representation chosen for the states, the intermediate variables and the coefficients.

They are modelled as independent white noise, characterized by their first and second order moments (section V).

Denote ξ the vector with all the added noise sources:

$$\xi(k) \triangleq \begin{pmatrix} \xi_T(k) \\ \xi_X(k) \\ \xi_Y(k) \end{pmatrix} \quad (16)$$

Proposition 1 *It is then possible to express the implemented system as the initial system with a noise $\xi'(k)$ added on the output(s) (see figure 1).*

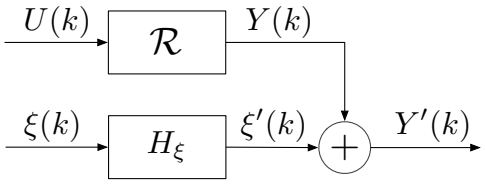

Fig. 1. Equivalent system, with noises extracted

$\xi'(k)$ is the noise $\xi(k)$ through the transfer function H_ξ defined by:

$$H_\xi : z \to C_Z \left(zI_n - A_Z\right)^{-1} M_1 + M_2 \tag{17}$$

with

$$M_1 \triangleq \begin{pmatrix} KJ^{-1} & I_n & 0 \end{pmatrix} \tag{18}$$

$$M_2 \triangleq \begin{pmatrix} LJ^{-1} & 0 & I_{p_2} \end{pmatrix} \tag{19}$$

Proof: H_ξ comes from a reformulation of equation (15), with ξ as one input of the system. ■

Proposition 2 *The output noise power is defined as the power of the noises added on the output(s):*

$$P \triangleq E\left\{\xi'(k){\xi'}^{\top}(k)\right\} \tag{20}$$

It is evaluated by:

$$P = tr\left(\psi_\xi \left(M_2^{\top} M_2 + M_1^{\top} W_o M_1\right)\right) + \mu_{\xi'}^{\top} \mu_{\xi'} \tag{21}$$

where $\mu_{\xi'} = (C_Z(I - A_Z)^{-1}M_1 + M_2)\mu_\xi$ and W_o is the observability grammian of the system $\mathcal{R}$.

Proof: Eq. (20) leads to $P = \sigma_{\xi'}^2 + \mu_{\xi'}^{\top}\mu_{\xi'}$. Then, lemma 1 is applied on noise $\xi(k)$ through H_ξ. ■

Remark 1 Equation (21) is a good illustration of the relationship between the works done in the *hardware/software* community and the one done in the *control* community: ψ_ξ and μ_ξ depend only on H/W implementation, whereas the other terms only depend on the algorithm used.

IV. Fixed-point Implementation

Since each variable can have free fixed-point format, it exists various fixed-point possible implementations for steps (i) to (iii). This section specifies implementation scheme and fix the fixed-point formats, in order to dertermine $\xi(k)$ and then ψ_ξ.

A. Fixed-point representation

In the paper, the notation (β, γ) is used for the fixed-point representation: β is the total wordlength in bits of the representation, whereas γ is the fractional part wordlength (see figure 2). These parameters could be scalars, vectors or matrices, according to the variables they refer to.

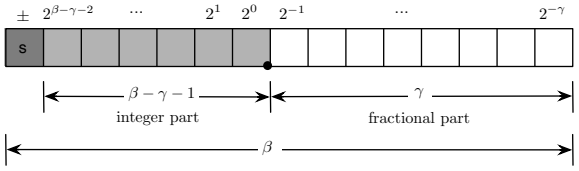

Fig. 2. Fixed-point representation

B. Scalar product

As the implementation of eq. (1) only requires scalar products, let us consider the following scalar product:

$$S = \sum_{i=1}^{n} P_i E_i \tag{22}$$

where $(P_i)_{1 \leqslant i \leqslant n}$ are given coefficients and $(E_i)_{1 \leqslant i \leqslant n}$ some bounded variables.
Let us consider the following very large hypotheses:

- (β, γ) is the fixed-point format of S, such that there is no overflow when evaluating $\sum P_i E_i$
- (β_i, γ_i) are the fixed-point format of E_i $(\forall 1 \leqslant i \leqslant n)$
- $(P_i)_{1 \leqslant i \leqslant n}$ are given, and represented with β_i' bits. The fixed-point format (β_i', γ_i') can be deduced by:

$$\gamma_i' \triangleq \beta_i' - 2 - \left\lfloor \log_2 |P_i| \right\rfloor \tag{23}$$

  where the $\lfloor x \rfloor$ operation rounds x to the nearest integer lower than or equal to x.

It is also considered that all the additions are realized by one accumulator with β_{ADD} bits, plus β_g guards bits and **that all the additions are done on the same fixed-point format $(\boldsymbol{\beta_{ADD} + \beta_g, \gamma_{ADD}})$.**

Since all the fixed-point formats are defined, some quantization operations (in order to adapt the wordlength and/or the fractional wordlength) could be necessary. Figure 3 shows the complete flow.

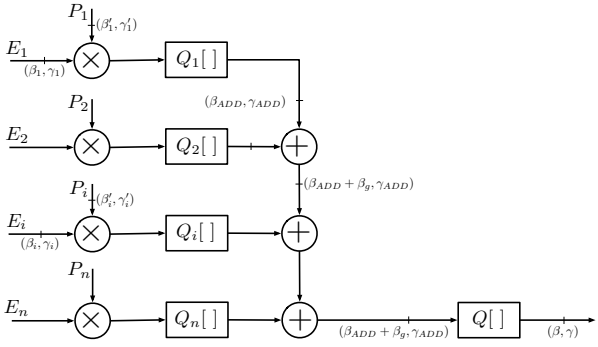

Fig. 3. Fixed-point formats of the scalar product

1) Accumulator fractional part determination: The common fixed-point format for the additions must be known. It is such that:

- the dynamic of each product can be represented without overflow. As $(\beta_i + \beta'_i, \gamma'_i + \gamma_i)$ is the format of each product, γ_{ADD} must satisfy ($\forall 1 \leqslant i \leqslant n$):

$$\beta_{ADD} - 1 - \gamma_{ADD} \geqslant \beta_i + \beta'_i - \gamma_i - \gamma'_i - 1 \quad (24)$$

- the final result may be represented without overflow with $(\beta_{ADD} + \beta_g, \gamma_{ADD})$ as format, so:

$$\beta_{ADD} + \beta_g - 1 - \gamma_{ADD} \geqslant \beta - 1 - \gamma \quad (25)$$

Then, the condition $\gamma_{ADD} \leqslant \underset{\max}{\gamma_{ADD}}$, with

$$\underset{\max}{\gamma_{ADD}} = \beta_{ADD} - \max\left(\beta - \beta_g - \gamma, \max_i \left(\beta_i + \beta'_i - \gamma_i - \gamma'_i\right)\right) \quad (26)$$

guaranties no overflow in the evaluation of S.

Moreover, γ_{ADD} should be the greatest possible in order to minimize the precision loss, but since $\gamma_{ADD} \geqslant \max_i (\gamma_i + \gamma'_i)$, there is no more precision loss.
So the following γ_{ADD} is chosen:

$$\gamma_{ADD} = \min\left(\underset{\max}{\gamma_{ADD}}, \max_i \left(\gamma_i + \gamma'_i\right)\right) \quad (27)$$

2) Quantization between product and accumulation: In order to align the fixed-point format after the products to the accumulator fixed-point format, there is a potential quantization block (shift) between each product and the accumulation. This scheme is called *Roundoff After Multiplication* (*RAM*).
It is also possible to move these shifts into the coefficients, by modifying their format such that the fixed-point format of each multiplication is compatible with the accumulator's one. In that case, the computational scheme is called *Roundoff Before Multiplication* (*RBM*).

So, let us introduce $\tilde{\gamma}'_i$ such that $(\beta'_i, \tilde{\gamma}'_i)$ is the final fixed-point format for the coefficient P_i:

$$\tilde{\gamma}'_i = \begin{cases} \gamma'_i & \text{in } \textit{RAM} \text{ scheme} \\ \gamma_{ADD} - \gamma_i & \text{in } \textit{RBM} \text{ scheme} \end{cases} \quad (28)$$

It is then necessary to add, after the i^{th} multiplication, a right shift of $\gamma_i + \tilde{\gamma}'_i - \gamma_{ADD}$ in order to align the formats (so, no shift in *RBM* case).

3) Final quantization: After the last addition in the accumulator, the result could be quantified to be stored in S. In order to align the accumulator format $(\beta_{ADD} + \beta_g, \gamma_{ADD})$ on the output format (β, γ), a right shift of $\gamma_{ADD} - \gamma$ bits is required.

C. Application to the Implicit Specialized Form

This analyze is applied to the Implicit Specialized Form. Let $\mathcal{R} = (Z, l, m, n, p)$ be a realization and let us consider that:

- the inputs $U(k)$ (bounded by $\overset{\max}{|U|}$) have (β_U, γ_U) as fixed-point format;
- the outputs, states and intermediate variables are represented with β_Y, β_X and β_T bits;
- the coefficients are represented with β_Z bits;
- the additions are done by accumulators of β_{ADD} bits, plus β_g guards bits ($\beta_{ADD} \in \mathbb{R}^{(l+n+m)\times 1}$).

Proposition 3 *The outputs, states and intermediate variables' formats are given by*[5]*:*

$$\begin{pmatrix} \gamma_T \\ \gamma_X \\ \gamma_Y \end{pmatrix} = \begin{pmatrix} \beta_T \\ \beta_X \\ \beta_Y \end{pmatrix} - 2.\mathbb{1}_{l+n+p,1} - \left\lfloor \log_2 \left(\|H_{\max}\|_{l_1} \overset{\max}{|U|} \right) \right\rfloor \quad (29)$$

where $\mathbb{1}_{k,l}$ represents the matrix of $\mathbb{R}^{k\times l}$ with all coefficients set to 1, $\|.\|_{l_1}$ the l_1-norm and

$$H_{\max} : z \to N_1 \left(zI_n - A_Z\right)^{-1} B_Z + N_2, \quad (30)$$

$$N_1 \triangleq \begin{pmatrix} J^{-1}M \\ I_n \\ C_Z \end{pmatrix}, N_2 \triangleq \begin{pmatrix} J^{-1}N \\ 0 \\ D_Z \end{pmatrix} \quad (31)$$

Proof: $H_{\max}$ is the transfer function from $U(k)$ to $T(k+1)$, $X(k+1)$ and $Y(k)$ and determine the maximum value for the intermediate variables, the states and the output(s). ■
Denote γ_Z the binary point position[6] of the coefficients Z:

$$\gamma_Z = \beta_Z - 2.\mathbb{1}_{l+n+p,l+n+m} - \lfloor \log_2 |Z| \rfloor \quad (32)$$

The fixed-point formats of the additions are given by:

$$\gamma_{ADD} = \beta_{ADD} - \max_{row} \left(\begin{pmatrix} \beta_T \\ \beta_X \\ \beta_Y \end{pmatrix} - \beta_g - \begin{pmatrix} \gamma_T \\ \gamma_X \\ \gamma_Y \end{pmatrix}, \alpha \right) \quad (33)$$

where

$$\alpha = \max_{row} \left(\beta_Z - \gamma_Z + \mathbb{1}_{l+n+p,1} \left(\begin{pmatrix} \beta_T \\ \beta_X \\ \beta_U \end{pmatrix} - \begin{pmatrix} \gamma_T \\ \gamma_X \\ \gamma_U \end{pmatrix} \right)^\top \right) \quad (34)$$

and $\max_{row}(M)$ returns a column vector with the maximum value of each row of M.
The final alignments are right shifts of d_{ADD} bits, with:

$$d_{ADD} = \gamma_{ADD} - \begin{pmatrix} \gamma_T \\ \gamma_X \\ \gamma_Y \end{pmatrix} \quad (35)$$

Denote $\tilde{\gamma}_Z$ the final binary point position of the coefficients Z, according to *RAM* or *RBM* scheme, and d_Z the shifts needed after each multiplication ($(d_Z)_{i,j}$ is the right shift needed after the multiplication by $Z_{i,j}$) in order to align the format after each multiplication. Then:

$$\tilde{\gamma}_Z = \begin{cases} \gamma_Z & \text{if } \textit{RAM} \\ \gamma_{ADD}.\mathbb{1}_{1,l+n+m} - \mathbb{1}_{l+n+p,1}. \begin{pmatrix} \gamma_T \\ \gamma_X \\ \gamma_U \end{pmatrix}^\top & \text{if } \textit{RBM} \end{cases} \quad (36)$$

[5] In order to simplify the expressions, matrix extensions of $\log_2$, floor operator $\lfloor . \rfloor$ and power of 2 are used. For example, if $M \in \mathbb{R}^{p\times q}$, then $\log_2(M) \in \mathbb{R}^{p\times q}$ such as $(\log_2(M))_{i,j} \triangleq \log_2(M_{i,j})$.

[6] $(\gamma_Z)_{i,j}$ could be $-\infty$ for null coefficients, but it is not a problem because such coefficients are not implemented

and

$$d_Z = \tilde{\gamma}_Z + \mathbb{1}_{l+n+p,1}.\begin{pmatrix}\gamma_T\\ \gamma_X\\ \gamma_U\end{pmatrix}^{\top} - \gamma_{ADD}.\mathbb{1}_{1,l+n+m} \quad (37)$$

(d_Z is a null matrix in *RBM* case).
With d_Z, $\tilde{\gamma}_Z$, γ_{ADD}, d_{ADD}, γ_T, γ_X and γ_Y, the fixed-point implementation of the controller is entirely defined.

V. Output noise power in *RBM* scheme

Due to lack of place, this section only focuses on *RBM* scheme (*RAM* case could be examined in similar way). In *Roundoff Before Multiplication* case, quantizations only occurs at the end of the additions, when the accumulator result is stored in intermediate variables, states or output(s) and a right shift of d_{ADD} bits is applied.

Remark 2 in *RBM* scheme, part of the roundoff errors is transformed in parametric errors, that are not studied here. See [5] for work on transfer function sensitivity.

The lemma 2 recalls the noise produced during shift:

Lemma 2 *Let $x(k)$ be a signal with fixed-point format $(\beta + d, \alpha + d)$. Right shifting $x(k)$ of d bits is similar to add to $x(k)$ the independent white noise $e(k)$.*
The right shift could round $x(k)$ towards $-\infty$ (truncation: default behaviour) or toward the nearest integer (nearest rounding: possible with some additional hardware/software operations [11]). If $d > 0$, the moments of $e(k)$ are given by:

	truncation	*best roundoff*
μ_e	$2^{-\gamma-1}(1-2^{-d})$	$2^{-\gamma-d-1}$
σ_e^2	$\frac{2^{-2\gamma}}{12}(1-2^{-2d})$	$\frac{2^{-2\gamma}}{12}(1-2^{-2d})$

(38)

else ($d \leqslant 0$) $e(k)$ is null.
Proof: See [2]. ∎

It is now possible to define the moments of $\xi(k)$:

Proposition 4 *Denote $\bar{\gamma} \triangleq \begin{pmatrix}\gamma_T\\ \gamma_X\\ \gamma_Y\end{pmatrix}$ and define s by*

$$s_i \triangleq \begin{cases}1 & \text{if } d_{ADDi} > 0\\ 0 & \text{otherwise}\end{cases} \quad (39)$$

Then μ_ξ is given by:

$$(\mu_\xi)_i = \begin{cases}s_i 2^{-\bar{\gamma}_i - 1} & \text{truncation}\\ s_i 2^{-\bar{\gamma}_i - 1 - d_{ADD}} & \text{nearest rounding}\end{cases} \quad (40)$$

and, since these noises are independent, ψ_ξ is diagonal with:

$$(\psi_\xi)_{i,i} = s_i \frac{2^{-2\bar{\gamma}_i}}{12}\left(1 - 2^{-d_{ADD}}\right) \quad (41)$$

Then, with equation 21, the roundoff noise power could be determined.

VI. Optimal Design

Since the Roundoff Noise power depends on the realization chosen to numerically realize the filter, it is of interest to find, among the equivalent realizations set, those with low roundoff noise.

In order to exploit the potential offered by the specialized implicit form in improving implementations, it is necessary to describe sets of equivalent system realizations. The *Inclusion Principle* introduced by Šiljak and Ikeda [8] in the context of decentralized control, could be extended to the Specialized Implicit Form in order to characterize equivalent classes of realizations [5]. Although this extension gives the formal description of equivalent classes, it is of practical interest to consider only realizations with the same dimensions, where transformation from one realization to another is only a similarity transformation.

Proposition 5 *Consider a realization $\mathcal{R}_0 = (Z_0, l, m, n, p)$. All realizations $\mathcal{R}_1 = (Z_1, l, m, n, p)$ such that*

$$Z_1 = \begin{pmatrix}\mathcal{Y} & & \\ & \mathcal{U}^{-1} & \\ & & I_p\end{pmatrix} Z_0 \begin{pmatrix}\mathcal{W} & & \\ & \mathcal{U} & \\ & & I_m\end{pmatrix} \quad (42)$$

are equivalent (with $\mathcal{U} \in \mathbb{R}^{n\times n}$, $\mathcal{Y} \in \mathbb{R}^{l\times l}$ and $\mathcal{W} \in \mathbb{R}^{l\times l}$ non-singular matrices).

For particular structured realizations, the transformation matrices $\mathcal{U}$, $\mathcal{Y}$ and $\mathcal{W}$ may be linked (for δ-state-space, $\mathcal{Y} = \mathcal{U}^{-1}$ and $\mathcal{W} = \mathcal{U}$, see (7) ; and for classical state-space, $\mathcal{Y} = \mathcal{W} = I_l$).

Then, the optimal design problem that consists in finding the best realization, among the equivalent realizations set, could be defined:

$$\mathcal{R}_{opt} = \underset{\mathcal{R}\in\mathscr{R}_H}{\arg\min} \mathcal{J}(\mathcal{R}) \quad (43)$$

where $\mathcal{J}$ is a FWL criteria (output roundoff noise gain, transfer function sensitivity [5], etc.) and $\mathscr{R}_H$ the set of equivalent realizations.

Due to the size of $\mathscr{R}_H$, this problem cannot be solved practically: the search is done among equivalent realizations with particular structure (state-space, cascade decomposition, sparse structures, δ-state-space, etc.).

Since the measure $\mathcal{J}$ could be non-smooth and/or non-convex, the Adaptive Simulated Annealing (ASA) method has been chosen to solve that problem ([9], [1]).

VII. Examples

To illustrate the roundoff noise power measure and the optimal design problem, let us consider the following lowpass Butterworth filter:

$$H(z) = \frac{0.003622z^2 + 0.007243z + 0.003622}{z^2 - 1.823z + 0.8372} \quad (44)$$

and the associated realizations in *Roundoff Before Multiplication* scheme, with input, output, coefficients, states, intermediate variables represented with 16 bits and 32 bits

for the accumulator (no guard bit). $\max|U| = 20$ (so $\gamma_U = 10$).
The following realizations are considered:

Z_1: direct form II with shift-operator,
Z_2: roundoff noise-optimal state-space realization,
Z_3: roundoff noise-optimal δ-realization (described by eq. (6)), with $\Delta = 2^{-5}$.

The roundoff noise-optimal realizations Z_2 and Z_3 are obtained by solving the optimal design problem for the output roundoff noise measure.
The roundoff noise power predicted by eq. (21) was confirmed by statistical measures on the difference between floating-point realizations and so-determined fixed-point realizations (with fixed-point quantized coefficients).

TABLE I
ROUNDOFF NOISE POWER AND COMPUTATIONAL COST

realization	roundoff noise power	Nb. operations
Z_1	$3.914e-3$	$4+\ 5\times$
Z_2	$3.903e-7$	$6+\ 9\times$
Z_3	$3.540e-7$	$8+\ 11\times$

The results are given in table I. Z_3 has the lowest roundoff noise power but required more operations than Z_1 and Z_2. The algorithm 1 presents the fixed-point algorithm associated to realization Z_3 (*RBM* scheme). If a low tolerance on roundoff noise is required, the simplest implementation Z_1 should not be used.
These results depend on the filter/controller. It is possible to find examples where the optimal state-space realization has a lower roundoff noise than the optimal δ-realization.

Even if it is not the goal of this paper, one could also consider the parametric errors (transfer function sensitivity, pole sensitivity). A global methodology with multi-objectives optimization will be presented in future papers.

Input: u: 16 bits integer
Output: y: 16 bits integer
Data: xn, xnp: array [1..2] of 16 bits integer
Data: Acc: 32 bits integer
begin
//Intermediate variables
$Acc \leftarrow (xn(1) * -26130)$;
$Acc \leftarrow Acc + (xn(2) * -26185)$;
$Acc \leftarrow Acc + (u * 8780)$;
$T0 \leftarrow Acc >> 15$;
$Acc \leftarrow (xn(1) * 8856)$;
$Acc \leftarrow Acc + (xn(2) * -10175)$;
$Acc \leftarrow Acc + (u * -24493)$;
$T1 \leftarrow Acc >> 15$;
//States
$Acc \leftarrow T0 << 12$;
$Acc \leftarrow Acc + xn(1) << 15$;
$xnp(1) \leftarrow Acc >> 15$;
$Acc \leftarrow T1 << 13$;
$Acc \leftarrow Acc + xn(2) << 15$;
$xnp(2) \leftarrow Acc >> 15$;
//Outputs
$Acc \leftarrow (xn(1) * 23633)$;
$Acc \leftarrow Acc + (xn(2) * 1808)$;
$Acc \leftarrow Acc + (u * 119)$;
$y \leftarrow Acc >> 15$;
//Permutations
$xn \leftarrow xnp$;
end
Algorithm 1: Numerical fixed-point algorithm of Z_3.

VIII. Conclusion

The Implicit State-Space Form provides a general framework for the analysis and design of digital filter implementation in FWL context. The fixed-point format determination for two computational schemes has been developed and exhibited, with a bit-accurate output noise power analysis. It allows to compare different realizations and to find the *optimal* one (according to roundoff noises). The fixed-point code can be automatically produced.
Our present work focuses on a global methodology to search for the *best* implementation and to solve tradeoff between sensitivity, roundoff errors, computational resources, etc.

References

[1] S. Chen and B.L. Luk. Adaptive simulated annealing for optimization in signal processing applications. *Signal Processing*, 79:117–128, 1999.
[2] G. Constantinides, P. Cheung, and W. Luk. Truncation Noise in Fixed-Point SFGs. *IEE Electronics Letters*, 35(23):2012–2014, Nov. 1999.
[3] M. Gevers and G. Li. *Parametrizations in Control, Estimation and Filtering Probems*. Springer-Verlag, 1993.
[4] T. Hilaire, P. Chevrel, and Y. Trinquet. Implicit state-space representation : a unifying framework for FWL implementation of LTI systems. In *Proc. of the 16th IFAC World Congress*. Elsevier, July 2005.
[5] T. Hilaire, P. Chevrel, and J.F. Whidborne. A unifying framework for finite wordlength realizations. *IEEE Trans. on Circuits and Systems*, 8(54), August 2007.
[6] T. Hilaire, D. Ménard, and O. Sentieys. Roundoff noise analysis of finite wordlength realizations with the implicit state-space framework. In *15th European Signal Processing Conference (EUSIPOC'07)*, September 2007.
[7] S.Y. Hwang. Minimum uncorrelated unit noise in state-space digital filtering. *IEEE Trans. on Acoust., Speech, and Signal Processing*, 25(4):273–281, August 1977.
[8] M. Ikeda, D. Šiljak, and D. White. An inclusion principle for dynamic systems. *IEEE Trans. Automatic Control*, 29(3):244–249, March 1984.
[9] L. Ingber. Adaptive simulated annealing (ASA): Lessons learned. *Control and Cybernetics*, 25(1):33–54, 1996.
[10] S. Kim, KI. Kum, and W. Sung. Fixed-point optimization utility for C and C++ based digital signal processing programs. *IEEE Transactions on Circuits and Systems*, 45(11):1455–1464, November 1998.
[11] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee. *DSP Processor Fundamentals: Architectures and Features*. Berkeley Design Technology, Inc, Fremont, CA, 1996.
[12] D. Ménard and O. Sentieys. Automatic evaluation of the accuracy of fixed-point algorithms. In *Proceedings of DATE02 (Design Automation and Test in Europe)*, march 2002.
[13] R. Middleton and G. Goodwin. *Digital Control and Estimation, a unified approach*. Prentice-Hall International Editions, 1990.
[14] C. Mullis and R. Roberts. Synthesis of minimum roundoff noise fixed point digital filters. In *IEEE Transactions on Circuits and Systems*, volume CAS-23, September 1976.
[15] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. Mc Graw Hill, 1991.