Analytical Fixed-Point Accuracy Evaluation in Linear Time-Invariant Systems

Daniel Menard, *Member, IEEE*, Romuald Rocher, *Member, IEEE*, and Olivier Sentieys, *Member, IEEE*

***Abstract*—One of the most important stages of floating-point to fixed-point conversion, is the evaluation of the fixed-point specification accuracy. This evaluation is required to optimize the data word-length according to accuracy constraints. Classical methods for accuracy evaluation are based on fixed-point simulations but they lead to very long optimization times.**

A new method based on an analytical approach is presented for the case of linear-time-invariant (LTI) systems. The use of this method in data word-length minimization processes reduces significantly the optimization time compared to simulation based methods. Our approach allows the automatic determination of the signal-to-quantization-noise-ratio (SQNR) expression at the system output according to the fixed-point data format. This method is valid for recursive and non-recursive LTI systems and takes account of the quantization modes (truncation or rounding). The theoretical concepts and the different methodology stages are explained. Then, the ability to efficiently evaluate the fixed-point specification accuracy is demonstrated through examples.

***Index Terms*—Accuracy evaluation, design automation, digital signal processing systems, fixed-point arithmetic, linear-time-invariant (LTI) systems, quantization noise, roundoff errors, signal-to-quantization-noise-ratio (SQNR).**

I. Introduction

EFFICIENT implementation of digital signal processing (DSP) applications in embedded systems requires the use of fixed-point arithmetic. Thus, the vast majority of embedded DSP applications are implemented in fixed-point architectures [1]–[4]. Indeed, fixed-point architectures are cheaper and more energy efficient than floating-point architectures. In the case of fixed-point systems, the memory and the bus word-lengths are lower, leading to a markedly lower cost and power consumption. In the case of digital signal processors (DSPs), fixed-point processors compute data using 16 bits while for floating-point processors, data are stored using 32 bits to be compatible with the IEEE-754 norm. Moreover, floating-point operators are more complex in their processing of the exponent and the mantissa. Thus, chip area and latency are more important than for fixed-point operators. In particular, floating-point addition is markedly more complex because of the de-normalization and normalization stages.

Thus, DSP applications are designed and simulated using floating-point data types but they are finally implemented with fixed-point architectures. The development time of fixed-point systems is longer due to fixed-point conversion which determines the application's fixed-point specification. Some experiments [5] have shown that the manual fixed-point conversion process can represent from 25% to 50% of the total development time. This manual fixed-point conversion is a time-consuming and error prone task. In a recent survey [6], the fixed-point conversion was identified as one of the most difficult aspect of implementing an algorithm on an FPGA. Thus, reducing time-to-market requires high-level development tools with automatic fixed-point conversion. Consequently, methodologies for the automatic determination of the fixed-point specification have been proposed [7]–[9].

The fixed-point conversion process is made up of two main stages. The first stage corresponds to the definition of the data binary-point position through the determination of the integer part word-length. For this, the data dynamic range is evaluated to obtain the extreme values which have to be represented. Then, the data word-lengths are determined from the definition for each data of the fractional part wordlength. The efficient application implementation in hardware architectures—Application Specific Integrated Circuit (ASIC), Field Programmable Gate Array (FPGA)—requires the minimization of the chip size and power consumption. Thus, for the implementation, the goal is to minimize the data wordlength as long as the computation accuracy is maintained. In the digital signal processing domain, the most commonly used criteria for the evaluation of fixed-point specification accuracy is the Signal-to-Quantization-Noise-Ratio (SQNR) [7], [10], [11]. The output error due to fixed-point arithmetic is assumed to be a random variable (noise) and the SQNR metric represents the ratio between the signal power and the noise power. Thus, the data wordlengths are minimized as long as the SQNR constraint is met. Achievement of this optimization process is based on the availability of a tool allowing precise evaluation from the system output SQNR.

Numerous models have been proposed to evaluate the fixed-point accuracy of classical DSP algorithms like filters [12], FFT [13] and LMS based adaptive filters [14]. Each model is specific to the algorithm and can not be generalized. This approach is interesting in the case of optimized fixed-point IP generation [15] but is limited to these specific algorithms. Thus, these models do not allow the automatic computation of the fixed-point accuracy of any DSP application.

From the last decade, different approaches have been proposed to automate the fixed-point conversion. Most of the available methodologies for the fixed-point accuracy determination are based on simulation [7], [16], [17] and commercial tools have been proposed [6], [18], [19]. However, they lead to very long execution times for data wordlength optimization.

Manuscript received June 11, 2007; revised February 8, 2008. First published April 18, 2008; current version published November 21, 2008. This paper was recommended by Associate Editor E. Rogers.

The authors are with the IRISA/INRIA Laboratory, University of Rennes I, Lannion F-22300, France (email: Daniel.Menard@irisa.fr; Romuald.Rocher@irisa.fr; Olivier.Sentieys@irisa.fr).

Digital Object Identifier 10.1109/TCSI.2008.923279

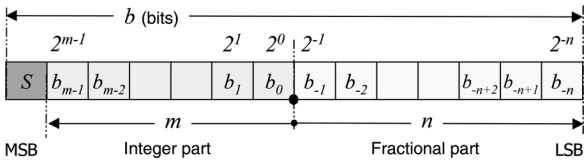


Fig. 1. Fixed-point data specification. The term b is the data wordlength and the terms m and n represent the number of bits respectively for the integer and the fractional part.

In this paper, an analytical method for SQNR evaluation in Linear-Time-Invariant (LTI) systems is presented. LTI systems represent the most important category of digital signal processing applications since they include digital filters and signal transformations (Fast Fourier Transform, Discrete Cosine Transform, Wavelet Transforms). The models used to obtain the output system SQNR according to the fixed-point specification are a generalization of the model proposed in [12]. The novelty of our approach lies in the ability to compute automatically the SQNR expression for any kind of linear-time invariant systems (non-recursive and recursive structures) from the signal flow graph representing the system. Moreover, a realistic noise model taking account of the quantization mode (truncation or rounding) is used. This method reduces significantly the wordlength optimization time compared to simulation based methods.

The paper is organized as follows. After an overview of the available methods for the SQNR evaluation, our approach is compared to the previous methodologies. In Section III, the theoretical concepts of our approach are explained. Then, the techniques used to implement this method and the associated CAD tool are detailed in Section IV. Finally, our method's ability to efficiently determine the SQNR expression is shown using some real examples in Section V.

II. Review of Fixed-Point Accuracy Evaluation Methods

Fixed-point data is made-up of an integer part and a fractional part as shown in Fig. 1. The fixed-point data format is specified as (b, m, n), where b, m and n are the number of bits (wordlength) respectively for the data, the integer part and the fractional part. In fixed-point arithmetic, m and n are fixed and lead to an implicit scale factor which does not change during the processing.

The aim of an automatic fixed-point conversion methodology is to define the integer and the fractional part wordlengths for each data and to include the scaling operations to obtain a correct fixed-point specification. The main parts of the methodology are shown in Fig. 2. The first stage is the data dynamic range determination. These results are used to define the minimal number of bits required for the data integer part to avoid an overflow during processing. The data wordlength is a tradeoff between the computation accuracy and the implementation efficiency. For a software implementation, the data wordlength can be optimized to minimize the execution time [9]. In this case, the data wordlengths take a limited number of values which depend on the data types supported by the processor. For a hardware implementation, the architecture is fully defined by the designer.

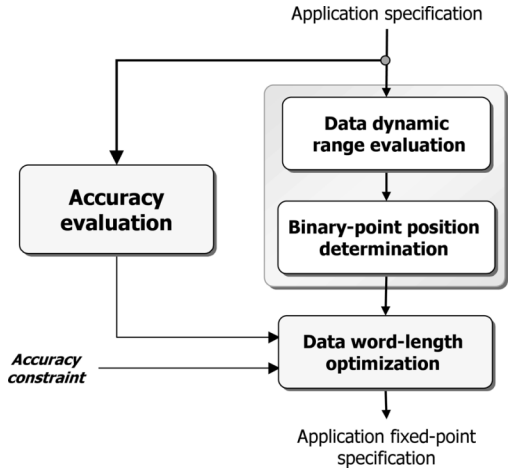


Fig. 2. Fixed-point conversion methodology. The fixed-point conversion is carried out under an accuracy constraint. The data wordlength optimization is based on an efficient accuracy evaluation tool.

Thus, the goal is to optimize the operator wordlengths to minimize the chip area and to ensure a minimal computation accuracy. Thus, the chip area is minimized as long as the accuracy constraint is fulfilled [11]. Given that the operator wordlengths are positive integers, the design space is very large. This optimization process requires the evaluation of the fixed-point accuracy specification many times. Thus, the accuracy evaluation process must be fast enough to obtain reasonable optimization times.

Simulation-based or analytical methods can be used to evaluate the SQNR at the system output. In the rest of this section, a review of the available methodologies is presented. The SQNR evaluation can be obtained with a bit-true simulation of the fixed-point application. The application bit-true simulation and the real execution give identical results at the bit level. This simulation can be done with system-level design tools such as CoCentric (Synopsys) [18] or Matlab-Simulink (Mathworks) [19]. Also, C++ classes to emulate the fixed-point mechanisms have been developed as in *SystemC* [18]. These techniques suffer from a major drawback which is the time required for the simulation [16]. It becomes a severe limitation when these methods are used in the data wordlength optimization process where multiple simulations are needed. The simulations are made on floating-point machines and the extra-code used to emulate fixed-point mechanisms increases the execution time to between one and two orders of magnitude compared to traditional simulations with floating-point data types [17]. To obtain an accurate estimation of the noise statistic parameters, a great number of samples must be taken for the simulation. For example, in [10], 10^5 input samples are used to simulate the algorithm and in [17] the authors proposed to use between 10^9 and 10^{12} samples. This large number of samples combined with the fixed-point mechanism emulation leads to very long simulation times. Different techniques [16], [20], [7] have been investigated to reduce this. Nevertheless, the data wordlength optimization process needs to explore the design-space of different data wordlengths. When simulation-based methods are used, this optimization is done with an iterative process where

the data wordlengths are progressively minimized while the accuracy constraint is met [7], [21], [10]. A new fixed-point simulation is required when the data wordlength is modified. Thus, the optimization time for a complex system becomes very long.

An alternative to the simulation based method is an analytical approach that determines the noise power expression at the system output according to the statistical parameters of the different noise sources induced by quantization. For this approach, two advantages can be emphasis. Firstly, this method gives an analytical SQNR expression and thus provides more information about the noise behavior in the system than a simulation based method which only gives the SQNR numerical value. Secondly, the execution time required to evaluate the noise power is definitely lower, especially for the data wordlength optimization process in hardware design. Indeed, the SQNR expression determination is done only once. Then, the SQNR evaluation corresponds to a mathematical expression computation.

Analytical expressions of the SQNR have been formulated for some particular DSP applications like digital filters in [12]. In [22], the authors have proposed an SQNR evaluation methodology based on an analytical approach for applications described with a data flow graph. The technique is based on the propagation of the noise statistical parameters in the graph. Thus, the application graph must be acyclic and consequently it can only be applied to non-recursive structures. Moreover, different restrictive assumptions have been made to define the operation output variance expressions. The operator input variables are considered to be independent and centred. Besides, the model does not take into account the signal power and the truncation quantization law is not supported. The signal and noise models are not realistic and the method is limited to non-recursive structures.

The method proposed by Shi in [23] is based on the perturbation theory which assumes that the quantization noise is a small deviation from the infinite precision signal. Let $x_1,\ldots,x_n$ be the n inputs of an operator and $b_{g_1},\ldots,b_{g_n}$ the noises associated with each input. The finite precision output is a function ϕ of the inputs and the noise terms. A second order Taylor development is used to obtain the finite precision output expression as defined in the following expression:

$$\begin{aligned}\phi_t(x_1,\ldots,x_n,b_{g_1},\ldots,b_{g_n}) &= \phi_t(x_1,\ldots,x_n,0,\ldots,0)+\sum_{i=1}^{n}\frac{\partial\phi_t}{\partial b_{g_i}}b_{g_i}\\ &+\sum_{i,j=1}^{n}\frac{\partial\phi_t}{\partial b_{g_i}b_{g_j}}b_{g_i}b_{g_j}. \end{aligned}\quad (1)$$

This model is applied to each operator. Then, the first- and second-order moments of the output quantization noise is computed. The following expression is obtained:

$$E(b_y^2)=\overline{\mu}^t B\overline{\mu}+\sum_{i=0}^{N_g'}c_i\sigma_{b_{g_i}}. \quad (2)$$

The term B is a N_g by N_g square matrix where N_g corresponds to the number of noise sources. The terms B and c_i are obtained statistically through simulations. The determination of each element of the matrix B and each element c_i requires a new simulation. The minimal number of simulations to obtain these terms is $\mathcal{O}(N_g^2)$. To have a general expression for the output noise power, a noise source must be considered for each operator input and output. Thus, the execution time to obtain the general expression for the output noise power can become important.

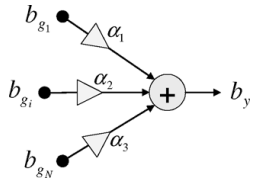


Fig. 3. Output quantization noise model in a fixed-point system. The system output noise b_y is a weighted sum of the different noise sources b_{g_i}.

In this paper a new method based on the analytical approach is proposed. It uses a realistic noise model which takes into account the different quantization modes (rounding and truncation). Moreover, this method allows the computation of the SQNR in non-recursive and recursive LTI systems. Our approach is based on the automatic computation of the system transfer functions from the graph representing the application.

III. Model for Linear Fixed-Point Systems

The aim of this section is to present the approach which allows us to obtain the SQNR expression of an LTI system output. The main challenge comes from the computation of the system output quantization power. In fixed-point systems, a quantization noise b_{g_i} is generated when some bits are eliminated during a fixed-point format conversion (cast operation). Each quantization noise source b_{g_i} is propagated inside the system and contributes to the output quantization noise b_y through the gain α_i as shown in Fig. 3. The analytical approach's goal is to define the output noise b_y power expression according to the noise source b_{g_i} parameters and the gains α_i between the output and the different noise sources.

In Section III-A, the noise models used in our approach are shown. This part defines the statistical parameters of the quantization noise sources and how the noises are propagated through the basic operators. Then, in Section III-B, the power expression of the system output quantization noise is defined according to the noise source statistical parameters.

A. Noise Models

1) Quantization Noise Models: In finite precision arithmetic, signal quantization leads to an unavoidable error. A commonly used model for the continuous-amplitude signal quantization, has been proposed by Widrow in [24] and refined in [25]. The quantization of the signal x is modeled by the sum of this signal and a random variable b_g. This additive noise b_g is a uniformly distributed white noise that is uncorrelated with the signal x and the other quantization noises. The validity conditions of these quantization noise properties have been defined in [25]. These conditions are based on the signal x characteristic function which is the Fourier transform of the probability density function (PDF). The authors have demonstrated that this model is valid when the signal x dynamic range is sufficiently greater than the quantum step size and the input bandwidth is large enough.

TABLE I
STATISTICAL PARAMETERS (MEAN μ_{b_z} AND VARIANCE $\sigma^2_{b_z}$) OF THE NOISE DUE TO THE QUANTIZATION OF THE SIGNAL z FOR THE DIFFERENT QUANTIZATION LAWS f_Q (R: ROUNDING, T: TRUNCATION).

f_Q	Statistical parameters	Continuous amplitude [24]	Discrete-amplitude $z = x \times y$, $z = x + y$ [27]	Discrete-amplitude $z = x \times C$ [28]
R	μ_{b_z}	0	$\frac{q}{2}(2^{-k})$	$\frac{q}{2}(2^{-(k-l)})$
	$\sigma^2_{b_z}$	$\frac{q^2}{12}$	$\frac{q^2}{12}(1-2^{-2k})$	$\frac{q^2}{12}(1-2^{-2(k-l)})$
T	μ_{b_z}	$\frac{q}{2}$	$\frac{q}{2}(1-2^{-k})$	$\frac{q}{2}(1-2^{-(k-l)})$
	$\sigma^2_{b_z}$	$\frac{q^2}{12}$	$\frac{q^2}{12}(1-2^{-2k})$	$\frac{q^2}{12}(1-2^{-2(k-l)})$

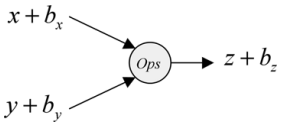


Fig. 4. Operation (*Ops*) noise model.

This model has been extended to model the computation noise in a system resulting from some bit elimination during a fixed-point format conversion. More especially, the round-off error resulting from the multiplication of a constant by a discrete amplitude signal has been studied by Barnes *et al.* in [26]. This study is based on the assumption that the probability density function (PDF) is continuous. However, this hypothesis is no longer valid when the number k of bits eliminated during a cast operation is small.

In [27], a model based on a discrete PDF is suggested and the quantization noise first and second-order moments are given. In this study, the probability value of each eliminated bit to be equal to 0 or 1, is assumed to be 1/2. This hypothesis is analyzed in detail in [28] and a new model is proposed for the case of the multiplication of a signal x by a constant C. In particular, for the constant C, the number l of consecutive zeros from the LSB is taken into account. For the discrete model, the additive noise can be considered as white noise that is uncorrelated with the signal and the other quantization noises.

In Table I, the quantization noise first and second order moments are summarized for the rounding and truncation quantization modes. The parameter q corresponds to the quantum step size and is equal to the weight associated with the least significant bit ($q = 2^{-n}$ where n is the fractional part wordlength).

2) Propagated Noise Models: In this part, the aim is to define the propagated noise models which define the operator output noise from the operator input noises. An operator with two inputs X and Y and one output Z is considered. The inputs X and Y and the output Z are made-up respectively of the signals x, y and z and the quantization noises b_x, b_y and b_z as presented in Fig. 4.

For an adder, the expressions of the output signal z and the output quantization noise b_z are

$$Z = X + Y \quad \Rightarrow \quad \begin{cases} z = x + y \\ b_z = b_x + b_y. \end{cases} \tag{3}$$

For multiplication the expressions of z and b_z are

$$Z = X \times Y \quad \Rightarrow \quad \begin{cases} z = xy \\ b_z = b_x y + b_y x + b_x b_y. \end{cases} \tag{4}$$

The term $b_x b_y$ represents the product of two quantization noises which is much smaller than the two other terms [24]. Indeed, the noise terms are considered to be smaller than the signal terms. Then, it is neglected in the following. If one of the inputs is a constant C such that Δ_C is a bias resulting from the quantization of the constant C and C' the constant value after the quantization of C, the expressions of z and b_z are given in (5). The bias Δ_C is not taken into account to evaluate the SQNR. The finite wordlength effect for the coefficient must be analyzed separately with other metric like transfer function sensitivity, distance to instability or frequency response modification.

$$Z = X \times C \Rightarrow \begin{cases} z = xC' \\ b_z = b_x C'. \end{cases} \tag{5}$$

B. Output Noise Power Expression

The output $y[n]$ of a single-input single-output (SISO) linear time-invariant (LTI) system is computed from the input $x[n]$ with the following expression [29]

$$y[n] = \sum_{p=0}^{N_p} b_p \cdot x[n-p] + \sum_{q=1}^{N_q} a_q \cdot y[n-q] \tag{6}$$

where, the $N_p + 1$ coefficients b_p and the N_q coefficients a_q are constant. If all the coefficients a_q are null, the system is non-recursive. Let $H(z)$ be the transfer function between the output $Y(z)$ and the input $X(z)$ obtained from (6) with the help of the $\mathcal{Z}$ transform [30]

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{p=0}^{N_p} b_p \cdot z^{-p}}{1 - \sum_{q=1}^{N_q} a_q \cdot z^{-q}}. \tag{7}$$

Let $h[n]$ be the system impulse response obtained from the inverse $\mathcal{Z}$ transform of the transfer function $H(z)$. This impulse

response allows to express the output $y[n]$ only from the input $x[n]$ as follows

$$y[n] = h[n] * x[n] = \sum_{k=-\infty}^{+\infty} h[k]x[n-k]. \tag{8}$$

A multiple-input single-output (MISO) linear time-invariant system made-up of N_e inputs $x_j[n]$ and one output $y[n]$ is considered. For a multiple-output system, our method is repeated for each output. Let $H_j(z)$ be the partial transfer function between the output $Y(z)$ and each input $X_j(z)$, and $h_j[n]$ be the impulse response associated with $H_j(z)$. The output $y[n]$ expression is equal to

$$y[n] = \sum_{j=0}^{N_e-1} h_j[n] * x_j[n]. \tag{9}$$

This system fixed-point version is detailed below. Let $\widehat{x_j}[n]$, be the j^{th} quantized system input and $\widehat{H_j}(z)$ be the transfer function between the output $Y(z)$ and the input $X_j(z)$ with quantized coefficients. The fixed-point arithmetic gives rise to an output computation error $b_y[n]$ which is defined as the difference between $y[n]$ and $\widehat{y}[n]$[1]. This output computation error is due to two kinds of error source. The noise $b_{ej}[n]$ results from the propagation in the system of the input quantization noise $b'_{ej}[n]$ associated with the input $\widehat{x_j}[n]$. When a cast operation occurs, some bits are eliminated and a quantization noise $b'_{gi}[n]$ is generated. The output noise resulting from the propagation of $b'_{gi}[n]$ is called $b_{gi}[n]$. Let $H_{gi}(z)$ be the transfer function between $B_{gi}(z)$ and $B'_{gi}(z)$. These two types of noise source are modelled by a uniformly distributed additive white noise as defined in the previous section. The output noise $b_y[n]$ final expression is equal to

$$b_y = \hat{y}[n] - y[n] = \underbrace{\sum_{j=0}^{N_e-1} h_j[n] * b'_{ej}[n]}_{b_e[n]} + \underbrace{\sum_{i=0}^{N_g-1} h_{gi}[n] * b'_{gi}[n]}_{b_g[n]}. \tag{10}$$

The system noise model is given in Fig. 5. This noise model is a generalization of the one proposed in [12].

1) Output Noise Power P_{b_y}: The approaches used to determine the expression of the statistical parameters of $b_{ej}[n]$ and $b_{gi}[n]$ are identical. Indeed, these noises represent the output of a linear subsystem (h_j, h_{gi}) having a white noise $(b'_{ej}[n], b'_{gi}[n])$ as input as defined in Section III-A-I.

The noise $b_{ej}[n]$ is the output of the linear subsystem with the transfer function $H_j(z)$ and input $b'_{ej}[n]$. The $b_{ej}[n]$ noise mean $(\mu_{b_{ej}})$ is obtained from the $b'_{ej}[n]$ input noise mean $(\mu_{b'_{ej}})$ and the transfer function $H_j(z)$ with the following expression

$$\mu_{b_{ej}} = E(b_{ej}) = \alpha_{ej} \cdot \mu_{b'_{ej}} = \mu_{b'_{ej}} H_j(1). \tag{11}$$

The noise power corresponding to the output noise $b_{ej}[n]$ second-order moment is equal to the auto-correlation function

[1]For the infinite and finite precision systems, the quantized coefficients are under consideration.

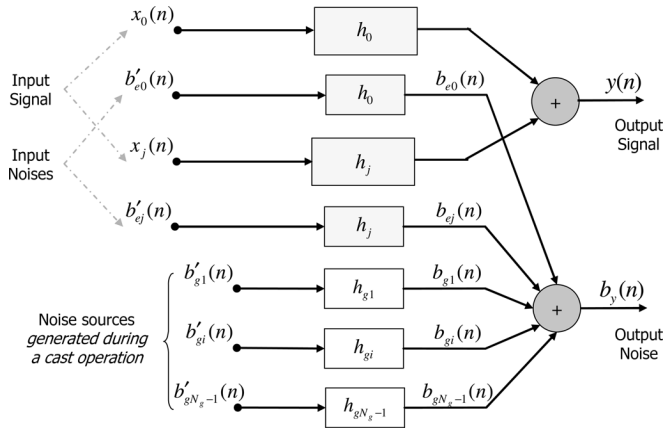


Fig. 5. Linear-time-invariant system noise model.

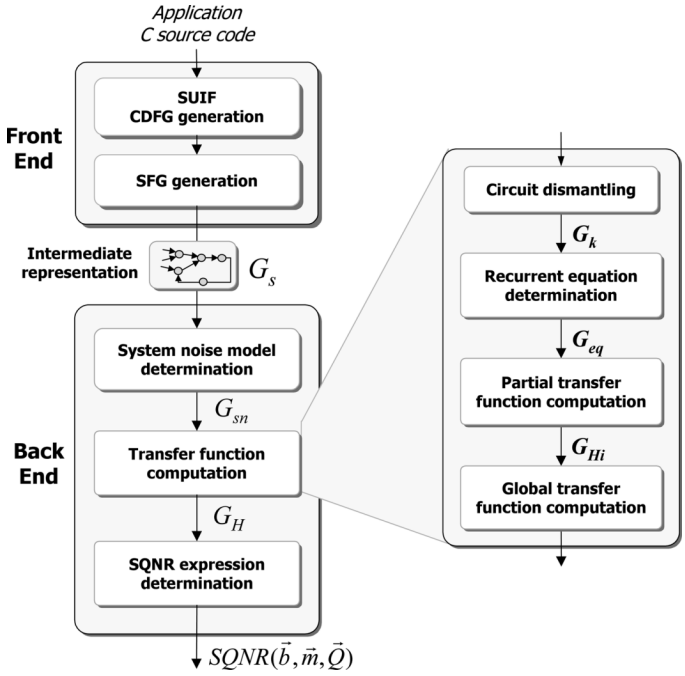


Fig. 6. Methodology for the SQNR expression determination.

of $b_{ej}[n]$ evaluated in 0. Using the Parseval identity, the noise power expression is obtained from the mean $\mu_{b'_{ej}}$ and the variance $\sigma_{b'_{ej}}$ of $b_{ej}[n]$ as follows

$$\sigma^2_{b_{ej}} = E(b^2_{ej}) - \mu^2_{b_{ej}} = \beta_{ej} \cdot \sigma^2_{b'_{ej}} = \sigma^2_{b'_{ej}} \cdot \frac{1}{2\pi} \int_{-\pi}^{\pi} |H_j(e^{j\Omega})|^2 d\Omega. \tag{12}$$

The noise $b_{gi}[n]$ statistical parameters can be obtained from the expression 11 and 12 by replacing b'_{ej} by b'_{gi} and H_j by H_{gi}.

The noise $b_y[n]$ is the sum of $N_e + N_g$ random variables $b_j[n]$. The $b_y[n]$ first order moment expression is equal to

$$E(b_y) = \sum_{j=0}^{N_e-1} \mu_{b_{ej}} + \sum_{i=0}^{N_g-1} \mu_{b_{gi}}. \tag{13}$$

From the noise model shown in Section III-A-I, each noise source $b'_j[n]$ is not correlated with a signal or another noise source. Thus, the output $b_j[n]$ of the linear system $h_j[n]$ with $b'_j[n]$ as input, will be correlated with no signal or no other noise

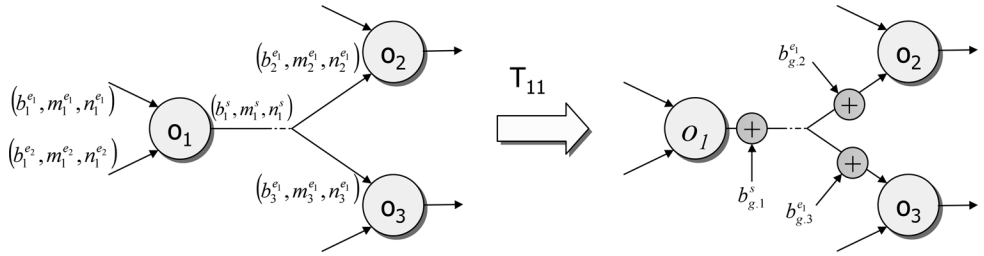


Fig. 7. Transformation T_1: generated noise source insertion.

source. Thus, the $b_y[n]$ second order moment expression is equal to

$$E(b_y^2) = \sum_{j=0}^{N_e-1} \sigma_{b_{ej}}^2 + \sum_{i=0}^{N_g-1} \sigma_{b_{gi}}^2 + \left(\sum_{j=0}^{N_e-1} \mu_{b_{ej}} + \sum_{i=0}^{N_g-1} \mu_{b_{gi}} \right)^2. \tag{14}$$

The output noise power expression is based on the computation of the different β_{ej} and α_{ej} terms. It requires the knowledge of the different transfer functions $H_j(z)$, $H_{g_i}(z)$ and the statistical parameters of the noise sources $b'_{ej}[n]$ associated with the input and the noise sources $b'_{gi}[n]$ generated inside the system. These different elements are automatically determined by the methodology presented in the next section.

IV. Method for SQNR Expression Determination

The aim of this method is to compute the application output SQNR expression by using an analytical method based on the theoretical approach presented above. This method is applied to the graph representing the application. To be independent of the application specification language, the tool, developed to implement this method, is split into two parts, a front-end and a back-end. The tool structure is shown in Fig. 6.

The front-end transforms the original application representation into a single graph, called G_s. The back-end determines the analytical SQNR expression. It consists of three successive transformations (T_1 to T_3) of the application graph, described in the following sections. First of all, the graph G_s is transformed into a graph G_{sn} representing the application at the quantization noise level. Next, the transfer functions between the output and all the inputs are evaluated.

Finally, the noise power expression is computed. The terms α_{ej} and β_{ej} are computed from the different transfer function frequency responses. The statistical parameters $\mu_{b'_{ej}}$ and $\sigma_{b'_{ej}}$ are computed from the data wordlength, the binary position and the quantization laws. These three kind of elements are the variables of the SQNR function. The noise power expression is available through a C function. Then, this C code is compiled and dynamically linked to the fixed-point conversion tool.

A. Intermediate Representation Generation

The intermediate representation G_s is a signal flow graph (SFG) [31]. A SFG is a data flow graph which includes the delay operations between the data. The nodes N_s of the directed graph $G_s = (N_s, E_s)$ represent either an operator or a data. The edges E_s give the relation between the data and the operators. The graph G_s specifies the behavior of the application at the fixed-point level. A fixed-point format (b_i, m_i, n_i) is associated with each operation o_i inputs and output. The wordlength b_i and the binary position m_i are considered to be variables. Moreover, the quantization mode Q_i used when a cast operation is achieved, must be specified. Thus, the SQNR expression, obtained with this tool, is a function of the vectors $\vec{b} = [b_1, b_2, \ldots, b_i, \ldots b_{N_o}]$, $\vec{m} = [m_1, m_2, \ldots, m_i, \ldots m_{N_o}]$ and $\vec{Q} = [Q_1, Q_2, \ldots, Q_i, \ldots Q_{N_o}]$ which define respectively, for the N_o system operations, the data wordlength, the binary-point position and the quantization mode.

The linear-time-invariant system representing the application is defined with a C algorithm. The front-end first step corresponds to this C code transformation into a Control Data Flow Graph (CDFG) [31]. It is generated from the intermediate representation obtained with the SUIF tool [32]. The CDFG is made-up of Control Flow Graphs (CFG) which represent the application control structures. Each computation core of a basic bloc control structure is represented with a Data Flow Graph (DFG). The second step transforms the CDFG into a Signal Flow Graph (SFG). Thus, the control structures are eliminated to obtain a Data Flow Graph (DFG). Then, the temporal information is analyzed to insert the delay operations.

B. Transformation T_1: System Noise Model Determination

The goal of transformation T_1 is to represent the application at the quantization noise level through the graph G_{sn}. The aim is to detect and to insert in the graph the two kinds of noise source defined in Section III-B.

The different potential generated noise sources $b_{g\cdot i}^{\mathrm{x}}$ are included in the graph as illustrated in Fig. 7. Let $(\widetilde{b}_1^s, \widetilde{m}_1^s, \widetilde{n}_1^s)$ be the output theoretical format of the operation o_1. It represents the output format obtained if no bit is eliminated. The first noise source $b_{g\cdot 1}^s$ is due to some o_1 output bits being eliminated during its computation. The number of eliminated bits k_1^s corresponds to the difference between the theoretical and the real fractional part wordlengths

$$k_1^s = \widetilde{n}_1^s - n_1^s. \tag{15}$$

The noise sources $b_{g\cdot 2}^{e_1}$ and $b_{g\cdot 3}^{e_1}$ are due to the cast operations carried out between the operation o_1 result and the operation o_2 and o_3 input operands. The number of bits eliminated $k_j^{e_1}$ is equal to

$$k_j^{e_1} = n_1^s - n_j^{e_1} \qquad \text{with } j = 2 \text{ or } 3. \tag{16}$$

These noise source statistical parameters are computed from the noise model presented in Table I according to the number k of bit eliminated, the data format and the quantization mode used.

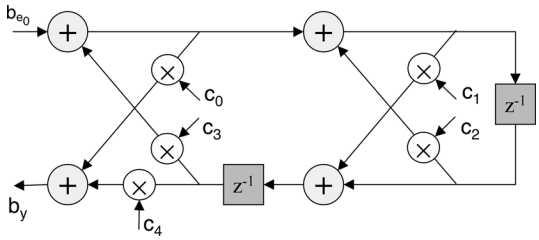


Fig. 8. Second order IIR filter implemented with a lattice structure. The z^{-1} nodes represent the delay operations between the data.

The second stage of the transformation leads to data and operator representation at the noise level. Each G_s data node is split into a *signal* node and a *noise* node. Then, each operator is replaced by its noise propagation model shown in Section III-A.

C. Transformation T_2: Transfer Function Computation

The goal of transformation T_2 is to determine the function set which defines the Linear-Time-Invariant (LTI) system. A method for determining the system transfer functions from a signal flow graph has been suggested in [33]. The transfer functions are determined through the system state matrix inversion. Nevertheless, according to the author, this approach does not allow the computation of the system transfer functions if it contains several identical poles. Moreover, structures like FIR filter (Finite Impulse Response) can not be processed because the algorithm used to obtain the transfer function requires that the transfer function poles have been determined.

In our approach, the transfer functions are obtained from the $\mathcal{Z}$ transform of the recurrent equations representing the system. It allows the processing of any kind of LTI system (non-recursive and recursive systems). The recurrent equations are built by traversing the graph from the inputs to the output. But, this technique is unusable if cycles are present in the graph as in our case when recursive structures are considered. Consequently, this technique requires first of all, the transformation of this graph into several directed a-cyclic graphs (DAG). The LTI system SFG is valid if each cycle contains at least a delay operation.

The different stages of the transformation T_2 are illustrated using an example corresponding to a second-order IIR (infinite impulse response) filter implemented with a lattice structure. The filter signal flow graph is shown in Fig. 8. To avoid complications only one noise source b_x, located at the filter input, is considered.

1) Transformation T_{21}: $G_{sn} \rightarrow G_k$: The T_{21} goal is to transform the graph G_{sn} into several directed acyclic graphs (DAG) G_k if G_{sn} contains cycles. Thus, for the first stage of transformation, the aim is to quickly detect a cycle occurrence in the graph and to decide if the transformation T_{21} has to be applied. To minimize this stage execution time, the cycle search procedure is carried out on the graph G_s and is stopped when a cycle is detected. To handle acyclic graphs with a large number of nodes, a cycle search algorithm based on a graph depth-first traversal is used. This algorithm allows allows the processing of each node only once.

In the T_{21} second stage, all the graph T_{21} cycles are enumerated with the algorithm proposed by Johnson [34]. The

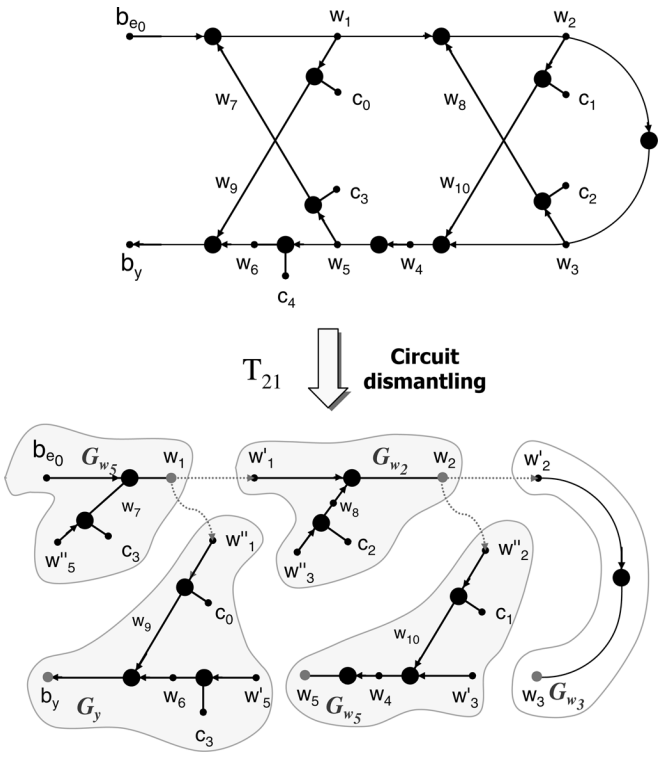


Fig. 9. Transformation T_{21} applied to the system presented in Fig. 8.

graph dismantling in DAG requires the definition of the points where the cycles have to be cut. After, enumerating a cycle $\mathcal{C} = \{w_j, \ldots, w_i \ldots, w_j\}$, all the paths L_i between the node w_j and the node representing the output are determined. The graph is dismantled at each node p_i corresponding to the last common data node between the cycle $\mathcal{C}$ and the i^{th} path L_i. The algorithm proposed by Johnson [34] has been modified to enumerate all the paths L_i between two nodes.

The cycle dismantling process is illustrated in Fig. 9 for the example presented in Fig. 8. Three cycles $\mathcal{C}_1$: $\{w_1, w_2, w_5, w_1\}$, $\mathcal{C}_2$: $\{w_1, w_2, w_3, w_5, w_1\}$ and $\mathcal{C}_3$: $\{w_2, w_3, w_2\}$ have been detected. The last common data node between the cycles $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ and all the paths having b_y as terminal node are w_1, w_2, w_3, w_5. The edges having the nodes w_i ($i = 1, 2, 3$ or 5) as their head are dismantled and the nodes w_i are duplicated. This transformation leads to four directed acyclic graphs (DAG) G_{w_1}, G_{w_2}, G_{w_3} and G_{w_5} as shown in Fig. 9. The duplicated nodes w_i' and w_i'' are sources (nodes without predecessor) of these new DAGs.

2) Transformation T_{22}: $G_k \rightarrow G_{\text{eq}}$: The graph $G_{\text{eq}} = (N_{\text{eq}}, E_{\text{eq}})$ is a weighted and directed graph which specifies the system with a recurrent equation set. This graph's nodes represent the system output, inputs and intermediate variables associated with the data nodes p_i. A weighted and directed edge (u, w, f_{wu}) from the node u to w indicates that the variable w is defined from the variable u with the recurrent equation f_{wu} specified through the edge weight.

The goal of transformation T_{22} is to build this graph G_{eq} from the different graphs G_w. The recurrent equation f_w associated with a DAG G_w is obtained by a DAG depth-first traversal. A post-order recursive algorithm is used to traverse the DAG. At

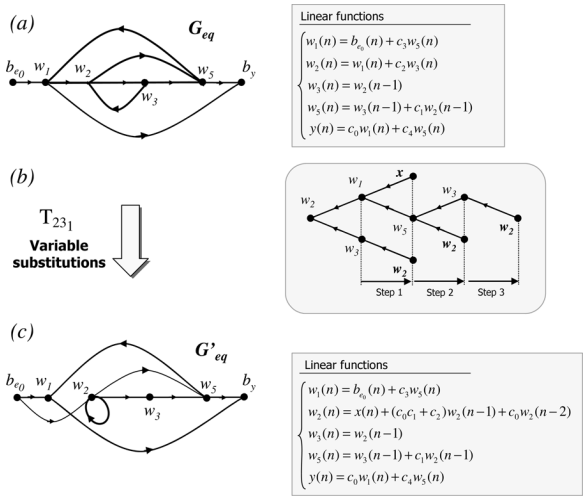


Fig. 10. Variable substitution example.

a node with predecessors, the algorithm examines these predecessors and then computes the recurrent equation from each predecessor and operator results. When a source is examined, the algorithm returns the name of the node i.e., the data name.

For the example specified in Fig. 8, the resulting graph G_{eq} and the obtained recurrent equations are presented in Fig. 10(a). The different nodes involved in this graph are the DAG sinks and sources: b_{e_0}, b_y, w_1, w_2, w_3 and w_5.

3) Transformation T_{23}: $G_{\text{eq}} \rightarrow G_{hi}$: The LTI system is defined with a partial transfer function set through the weighted and directed graph $G_{Hi} = (N_{G_{Hi}}, E_{G_{Hi}})$. The graph G_{Hi} nodes are members of the nodes N_{eq} set. The directed edge $(u, w, H_{wu}(z))$ from u to w is annotated with the partial transfer functions $H_{wu}(z)$ between the variable $W(z)$ and $U(z)$ respectively associated with the head w and the trail u of this edge

$$H_{wu}(z) = \frac{W(z)}{U(z)}. \tag{17}$$

The goal of transformation T_{23} is to compute these partial transfer functions from the recurrent equations with the technique shown at this end of the part.

A subsystem transfer function can be directly computed from the $\mathcal{Z}$ transform of the linear function defining the output w if this linear function contains explicitly all the terms related to the delayed versions of w. Thus, if any input to this subsystem is a function of the output w, variable substitutions must be made before the transfer function determination. Consequently, this restriction implies the computation of the $\mathcal{Z}$ transform of the recurrent equations on a graph G'_{eq} containing no cycle of length greater than one. Thus, the transformation T_{23} first step corresponds to the application of a set of variable substitutions to obtain this graph G'_{eq}. The variable substitution process allows independence of the way that the graph G_{sn} is dismantled and thus leads to the transfer function determination.

Let w_i be a node member of a cycle and S_{w_i} be the subgraph containing all the paths or the cycles having the variable

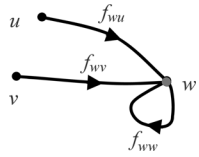


Fig. 11. G'_{eq} graph example.

w_i as terminal node. The initial node of each path is the first node which is not a cycle member. The substitution process is achieved with a subgraph S_{w_i} depth-first traversal. This technique leads to a solution if the subgraph S_{w_i} is a directed acyclic graph. Thus, the edges having w_i as their head are dismantled and this process must allow the dismantling of all the cycles contained in the subgraph S_{w_i}. Consequently, the chosen node w_i is the node which is common to the different cycles contained in S_{w_i}.

This variable substitution process is illustrated using the example presented in Fig. 10. The graph G'_{eq} in Fig. 10(a) is made-up of three cycles $\mathcal{C}_1 : (w_1, w_2, w_5, w_1)$, $\mathcal{C}_2 : (w_1, w_2, w_3, w_5, w_1)$ and $\mathcal{C}_3 : (w_2, w_3, w_2)$. The common node of the three cycles is w_2. The edges having w_2 as their head are dismantled. Then, the acyclic directed graph with w_2 as sink is depth-first traversed to apply variable substitutions as presented in Fig. 10(b). The graph sources are the node associated with the input b_{e_0} and the nodes corresponding to the duplication of w_2 carried out during the circuit dismantling process. The recurrent equation to compute the variable $w_2[n]$ is made up of only the variable $b_{e_0}[n]$ and the $w_2[n]$ delay version. The obtained graph G'_{eq} containing no cycle except loops is presented in Fig. 10(c).

The second step of the transformation T_{23} corresponds to the computation of the partial transfer functions from the graph G'_{eq}. They are obtained by the $\mathcal{Z}$ transform of the linear functions. To illustrate this process, the example shown in Fig. 11 is considered. This directed graph G'_{eq} is made-up of two sources u and v and one sink w. A loop is associated with the node w and annotated with the recurrent equation f_{ww}. From Mason's rule [35], the transfer function H_{wu} and H_{wv} between the output w and the inputs u and v are obtained with the help of the $\mathcal{Z}$ transform as follows:

$$H_{wu}(z) = \frac{W(z)}{U(z)} = \frac{\mathcal{Z}(f_{wu})}{1 - \mathcal{Z}(f_{ww})} \tag{18}$$

$$H_{wv}(z) = \frac{W(z)}{V(z)} = \frac{\mathcal{Z}(f_{wv})}{1 - \mathcal{Z}(f_{ww})}. \tag{19}$$

4) Transformation T_{24}: $G_{H_i} \rightarrow G_H$: The graph $G_H = (N_{G_H}, E_{G_H})$ is a weighted directed acyclic graph which specifies the system with a global transfer function set between the output and each system input. This DAG represents the system model shown in Fig. 5. In the graph G_{H_i}, the path between each system input and the output are determined and the global transfer functions are computed as a combination of the partial transfer functions of this path.

For the example considered in Fig. 8, the graph G_{H_i} obtained with the transformation T_{23} is shown in Fig. 12. This DAG sink is the node b_y and the source is the node b_{e_0}. Thus, the global

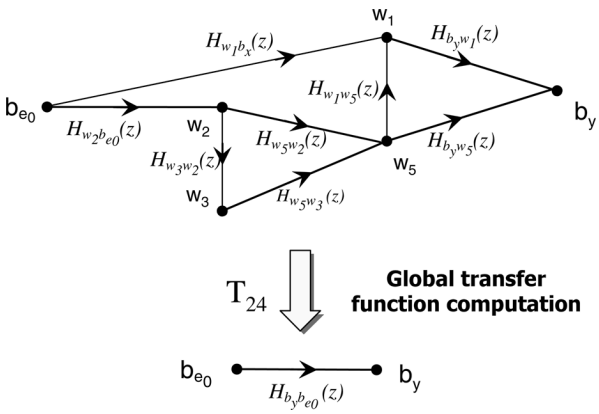


Fig. 12. Computation example of global transfer functions.

transfer function $H_{b_y b_{e_0}}(z)$ is determined from the different partial transfer functions as expressed in (20). This expression is obtained from all the paths between b_{e_0} and b_y.

$$\begin{aligned} H_{b_y b_{e_0}}(z) = {} & H_{b_y w_1}(z).H_{w_1 b_{e_0}}(z) \\ & + [H_{w_5 w_3}(z) \cdot H_{w_3 w_2}(z) H_{w_2 b_{e_0}}(z) \\ & + H_{w_5 w_2}(z) \cdot H_{w_2 b_{e_0}}(z)] \\ & \cdot [H_{b_y w_1}(z) \cdot H_{w_1 w_5}(z) + H_{b_y w_5}(z)]. \end{aligned} \quad (20)$$

The following global transfer function between the output b_y and the input b_{e_0} is obtained from (20)

$$H_{b_y b_{e_0}}(z) = c_0 + \frac{(c_0 c_1 c_3 + c_1 c_4) z^{-1} + (c_0 c_3 + c_4) z^{-2}}{1 - (c_2 + c_1 c_3) z^{-1} - c_3 z^{-2}}. \quad (21)$$

D. Transformation T_3 : SQNR Computation

The output noise power expression is computed from (14). Thus, the frequency responses of the different subsystems are computed from the transfer functions obtained after the transformation T_2. The output signal power required for the SQNR evaluation is computed from the system transfer functions and the system input parameters.

V. SQNR Computation Results

A. Quality of the Noise Power Estimation

Our approach allows the estimation of the SQNR from the fixed-point specification. To emphasise its quality, the precision of the SQNR estimation has been evaluated. In the SQNR expression, only the quantization noise power depends on the fixed-point specification. The signal power is a constant. Thus, the quality of our estimation depends only on the accuracy of the quantization noise power estimation. This quality has been evaluated through the measurement of the relative error Er between our analytical estimation P_{b_y} and the estimation $P_{b_y}^{(sim)}$ based on fixed-point simulations which is considered to be the reference. The relative error Er is defined using (22). For the simulation based estimation, the output quantization noise is obtained from the difference between the system outputs obtained with a fixed-point and a floating-point simulation. The floating-point simulation which uses double-precision types is considered to be the reference. Indeed, in this case, the error due to the floating-point arithmetic is definitely lower than the error due to the fixed-point arithmetic. Thus, the floating-point arithmetic errors can be neglected.

$$E_r = \left| \frac{P_{b_y} - P_{b_y}^{(sim)}}{P_{b_y}^{(sim)}} \right|. \quad (22)$$

The results obtained for several non-recursive LTI systems are shown in Fig. 13. The real correlator computes the correlation between a signal and a code of length N_c. The time-invariant code coefficients are called c_i. For the complex correlation application the signal and the code are complex. The *rake receiver* is a symbol estimation algorithm for third generation radiocommunication systems. For each application, different fixed-point specifications have been tested and each point represented in Fig. 13 corresponds to the relative error measured for a given fixed-point specification. The different fixed-point specifications are obtained by modifying the multiplication and the addition operand wordlength. The relative error maximal value is lower than 11% and on the 44 estimations obtained in this example, only two estimations are greater than 4%. A relative error of 11% is equal to an estimation error[2] Δ_{bit} of 0.15 bits.

For LTI recursive structures, different infinite-impulse response (IIR) filters have been tested. The results obtained for IIR filters implemented with a cascaded form are shown in Fig. 14 for different filter order. The relative error maximal value is lower than 3.3%, and on the 16 estimations, only one estimation is greater than 1.5%.

The results presented in Figs. 13 and 14, show that our estimations are accurate for recursive and nonrecursive LTI systems.

B. Tool Execution Time

To underline the efficiency of our approach in minimizing the fixed-point optimization time, the analytical and the simulation based approaches are compared. The time to evaluate the computation accuracy during the optimization process is measured for the two methods. The fixed-point simulations are done with SystemC. The results obtained for a rake receiver [36] used in the third generation telecommunication systems are shown in Fig. 15. The execution time T for evaluating the accuracy is measured according to the number of iterations i of the fixed-point optimization process. These two approaches have been tested on the same machine (Sun server).

For the analytical technique, the fixed-point accuracy is evaluated in two steps. Firstly, the SQNR expression is determined before the optimization process and this stage is only done once. In our example, the time to determine the SQNR expression is equal to 56 s. Secondly, during the optimization process, the accuracy evaluation corresponds to a mathematical expression computation. In our example, the time to evaluate the mathematical expression is equal to 33 μs. For the simulation based

[2]Let consider the quantization of one signal of wordlength b. This quantization leads to a noise power of $P(b)$. The term $\Delta_{bit}\left(E_{r_0}\right)$ is equal to $b_2 - b_1$ such as $(P(b_1) - P(b_2)/P(b_1)) = E_{r_0}$

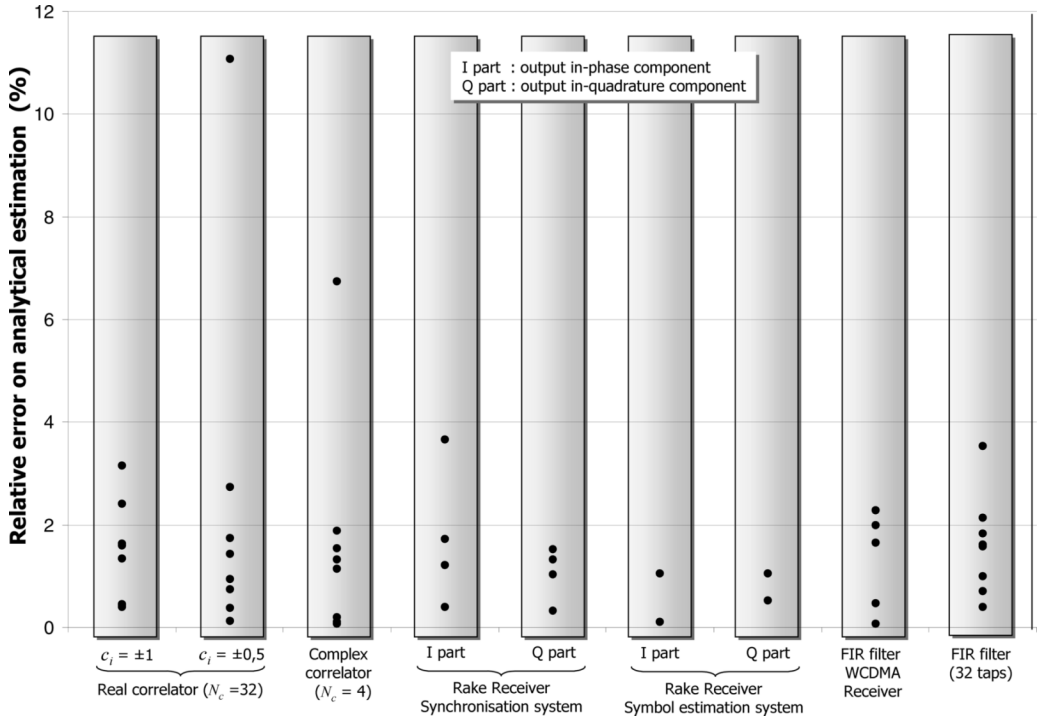


Fig. 13. Relative error on P_{b_y} estimation for non-recursive structures.

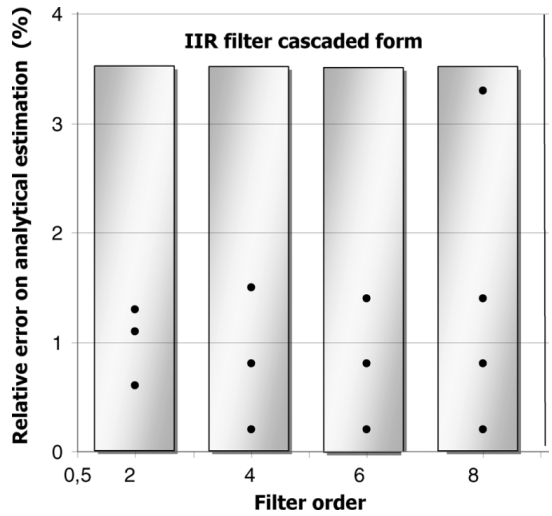


Fig. 14. Relative error on P_{b_y} estimation for different IIR filters (cascaded form) with different orders.

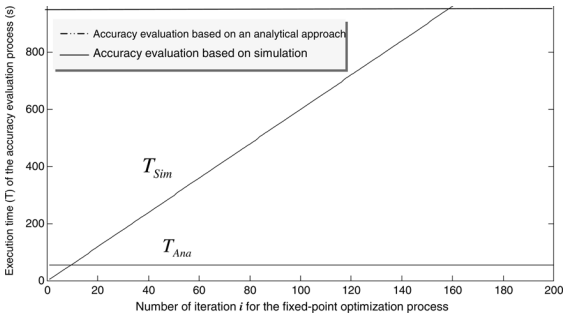


Fig. 15. Execution time T (s) of the accuracy evaluation according to the number of iterations i of the fixed-point optimization process.

approach, a new simulation is carried out for each accuracy evaluation. Thus, the optimization time is directly proportional to the number of iterations for the optimization process. As shown in Fig. 15, this approach leads to very long optimization times when the number of iterations becomes important. The time required to simulate the fixed-point specification with 5×10^4 input samples is equal to 5.2 s. The expressions of the execution time T to evaluate the accuracy for the analytical (T_{ana}) and the simulation based (T_{sim}) approaches are given in (23).

$$T_{\text{ana}}(i) = 56 + 3.3\,10^{-5} \times i \qquad T_{\text{sim}}(i) = 5.2 \times i. \quad (23)$$

Thus, our approach leads to better results when the number of iterations for the optimization process is greater than 10. For example, this application fixed-point specification has been optimized for a software implementation with the method shown in [9]. In this case, around 10^5 iterations are carried out to obtain an optimized solution for an optimization problem made-up of only 30 variables. With our approach the optimization time is equal to 89.3 s and with a simulation based approach the optimization time would be equal to 5.2×10^5 s. Thus, our approach leads to a definitively lower execution time compared to a simulation based approach to optimize the fixed-point specification.

VI. Conclusion

A methodology to determine an application output SQNR expression based on an analytical approach has been presented. The quality and the capability of our approach to compute the SQNR efficiently have been shown through different examples. This approach provides a significant improvement compared

to the simulation based methods for LTI systems. With our method, the time required to minimize the data wordlength is definitively lower. It allows a complete exploration of the design space and the determination of an optimized solution.

The techniques presented in this paper automatically determine the transfer functions from a signal flow graph. They can also be used for the data dynamic range estimation in a LTI system. Indeed, the dynamic range evaluation based on the l_1 or Chebychev norms [37] uses the transfer function concept. The tool for dynamic range estimation is under development.

REFERENCES

[1] B. Evans, "Modem design, implementation, and testing using NI's LabVIEW," in *Nat. Instrument Acad. Day*, Beirut, Lebanon, Jun. 2005.

[2] J. Eyre and J. Bier, "The evolution of DSP processors," *IEEE Signal Process. Mag.*, vol. 17, no. 2, pp. 43–51, Mar. 2000.

[3] W. Strauss, "DSP chips take on many forms," *DSP-FPGA.Com. Mag.*, Mar. 2006.

[4] J. Eyre and J. Bier, "DSPs court the consumer," *IEEE Spectrum*, vol. 36, no. 3, pp. 47–53, 1999.

[5] M. Clark, M. Mulligan, D. Jackson, and D. Linebarger, "Accelerating fixed-point design for MB-OFDM UWB systems," *CommsDesign*, Jan. 2005.

[6] T. Hill, "Acceldsp synthesis tool floating-point to fixed-point conversion of matlab algorithms targeting fpgas," in *White Papers*. : Xilinx, 2006.

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

[8] M. Willems, H. Keding, T. Grotker, and H. Meyr, "FRIDGE: An interactive fixed-point code generation environment for HW/SW-CoDesign," in *1997 IEEE Int. Conf. on Acoust., Speech and Signal Process. (ICASSP 97)*, Munich, Germany, Apr. 1997, pp. 297–290.

[9] D. Menard, D. Chillet, and O. Sentieys, "Floating-to-fixed-point conversion for digital signal processors," *EURASIP J. Appl. Signal Process.*, vol. 2006, pp. 1–19, Jan. 2006.

[10] H. Keding, F. Hurtgen, M. Willems, and M. Coors, "Transformation of floating-point into fixed-point algorithms by interpolation applying a statistical approach," in *9th Int. Conf. on Signal Proc. Appl. Technol. (ICSPAT 98)*, 1998.

[11] K. Kum and W. Sung, "Word-length optimization for high level synthesis of digital signal processing systems," in *IEEE Workshop on Signal Process. Syst. (SIPS 98)*, Boston, US, Oct. 1998, pp. 142–151.

[12] B. Liu, "Effect of finite word length on the accuracy of digital filters – A review," *IEEE Trans, Circuit Theory*, vol. CT-18, no. 6, pp. 670–677, Nov. 1971.

[13] Tran-Thong and B. Liu, "Fixed-point fast Fourrier transform error analysis," *IEEE Trans. Acoust. Speech, Signal Process.*, vol. ASSP-24, no. 6, pp. 563–576, Dec. 1976.

[14] C. Caraiscos and B. Liu, "A roundoff erro analysis of the LMS adaptive algorithm," *IEEE Trans. Acoust. Speech, Signal Process.*, vol. ASSP-32, no. 1, pp. 34–41, Feb. 1984.

[15] R. Rocher, D. Menard, N. Herv, and O. Sentieys, "Fixed-point configurable hardware components," *EURASIP J. Embedded Syst.,*, vol. 2006, no. Article ID 23197, pp. 1–13, 2006.

[16] L. D. Coster, M. Ade, R. Lauwereins, and J. Peperstraete, "Code generation for compiled bit-true simulation of DSP applications," in *IEEE Int. Symp. on Syst. Synthesis (ISSS'98)*, Hsinchu, Taiwan, Dec. 1998, pp. 9–14.

[17] H. Keding, M. Willems, M. Coors, and H. Meyr, "FRIDGE: A fixed-point design and simulation environment," in *IEEE/ ACM Conf. on Design, Automation and Test in Europe 1998 (DATE 98)*, Paris, France, Mar. 1998, pp. 429–435.

[18] F. Berens and N. Naser, *Algorithm to System-on-Chip Design Flow That Leverages System Studio and SystemC 2.0.1*. city??: Synopsys Inc., 2004.

[19] Fixed-Point Blockset User's Guide2.0 ed. Mathworks, 2001.

[20] M. Coors, H. Keding, O. Luthje, and H. Meyr, "Fast bit-true simulation," in *IEEE/ACM Des. Autom. Conf. 2001 (DAC 01)*, Las Vegas, US, Jun. 2001, pp. 708–713.

[21] W. Sung and K. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Trans. Signal Process.*, vol. 43, pp. 3087–3090, Dec. 1995.

[22] J. Toureilles, C. Nouet, and E. Martin, "A study on discrete wavelet transform implementation for a high level synthesis tool," in *IX European Signal Process. Conf. (EUSIPCO 98)*, Rhodes, Greece, Sep. 1998, pp. 459–461.

[23] C. Shi and R. Brodersen, "A perturbation theory on statistical quantization effects in fixed-point DSP with non-stationary input," in *IEEE Int. Symp. Circuits Syst. (ISCAS 04)*, Vancouver, Canada, 2004.

[24] B. Widrow, "Statistical analysis of amplitude quantized sampled-data systems," *Trans. AIEE Appl. and Ind.*, vol. 79, pt. II, pp. 555–568, 1960.

[25] A. Sripad and D. L. Snyder, "A necessary and sufficient condition for quantization error to be uniform and white," *IEEE Trans. Acoust. Speech, Signal Process.*, vol. ASSP-25, no. 5, pp. 442–448, Oct. 1977.

[26] C. Barnes, B. N. Tran, and S. Leung, "On the statistics of fixed-point roundoff error," *IEEE Trans. Acoust. Speech, Signal Process.*, vol. ASSP-33, no. 3, pp. 595–606, 1985.

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

[28] D. Menard, "Methodologie de Compilation D'Algorithmes de Traitement Du Signal Pour Les Processeurs en Virgule Fixe, Sous Contrainte de Precision," Ph.D. dissertaion, Univ. de Rennes I, Lannion, France, 2002.

[29] A. Oppenheim and R. Schafer, *Discrete Time Signal Processing*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 1999.

[30] K. Steiglitz, "Mathematical foundations of signal processing," in *Handbook for Digital Signal Processing*, S. Mitra and J. Kaiser, Eds. : Wiley, 1993, pp. 57–100.

[31] V. Madisetti, *VLSI Digital Signal Processors, an Introduction to Rapid Prototyping and Design Synthesis*. New York: IEEE Press, 1995.

[32] R. Wilson, "SUIF: An infrastructure for research on parallelizing and optimizing compilers," Stanford University, Computer Systems Laboratory, Tech. Rep. CA 94305-4055, 1994.

[33] L. Turner, D. Graham, and P. Denyer, "The analysis and implementation of digital filters using a special purpose CAD tool," *IEEE Trans. Edu.*, vol. 32, pp. 287–297, Aug. 1989.

[34] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM J. Computing*, vol. 4, no. 1, pp. 77–84, Mar. 1975.

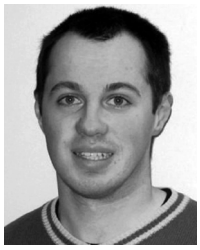
[35] S. Mason and H. J. Zimmermann, *Electron. Circuits, Signals and Syst.*. New York: Wiley, 1960.

[36] T. Ojanper and R. Prasad, *WCDMA : Towards IP Mobility and Mobile Internet*. Reading, MA: Artech House, 2002, Universal Personal Communications Series.

[37] T. Parks and C. Burrus, *Digital Filter Design*. New York: Wiley, 1987.

Daniel Menard received the Engineering degree and M.Sc. degrees in electronics and signal processing engineering from the University of Nantes Polytechnic School, Nantes, France, in 1996, and the Ph.D. degree in signal processing and telecommunications from the University of Rennes, France, in 2002.

From 1996 to 2000, he was a Research Engineer at the University of Rennes. He is currently an Associate Professor of Electrical Engineering at the University of Rennes (ENSSAT), Rennes, France, and a member of the CAIRN (Computing ArchItectures embedding Reconfigurable resources for eNergy-efficient systems-on-chip) research team at the IRISA/INRIA Laboratory. His research interests include implementation of signal processing and mobile communication applications in embedded systems, floating-to-fixed-point conversion, low power architectures and arithmetic operator design.



Romuald Rocher received the Engineering degree and M.Sc. degree in electronics and signal processing engineering from ENSSAT, University of Rennes, in 2003, the Ph.D. degree in signal processing and telecommunications from the University of Rennes, in 2006.

He is currently an Assistant Professor of Electrical Engineering at the University of Rennes and a member of the CAIRN (Computing ArchItectures embedding Reconfigurable resources for eNergy-efficient systems-on-chip) research team at the IRISA/INRIA Laboratory. His research interests include floating-to-fixed-point conversion, adaptive filters and architecture design for telecommunication systems.

Olivier Sentieys received the Engineering degree and M.Sc. degree in electronics and signal processing engineering from ENSSAT, University of Rennes, in 1990, the Ph.D. degree in signal processing and telecommunications from the University of Rennes, in 1993, and the "Habilitation à Diriger des Recherches" degree in 1999. He is currently a Professor of Electrical Engineering at the University of Rennes (ENSSAT).

He is the head of the CAIRN (Computing ArchItectures embedding Reconfigurable resources for eNergy-efficient systems-on-chip) research team at the IRISA/INRIA Laboratory and is a cofounder of Aphycare Technologies, a company developing smart sensors for biomedical applications. His research interests include VLSI integrated systems for mobile communications, finite arithmetic effects, low power and reconfigurable architectures, and multiple-valued logic circuits. He is the author or co-author of over 70 journal publications or published conference papers and holds four patents.