Automatic Evaluation of the Accuracy of Fixed-point Algorithms

Daniel Menard †
† LASTI - University of Rennes I
6, rue de kerampont
22300 Lannion, FRANCE
menard@enssat.fr

Olivier Sentieys †‡
‡ IRISA/INRIA
Campus de Beaulieu
35042 Rennes cedex, FRANCE
sentieys@irisa.fr

Abstract

The minimization of cost, power consumption and time-to-market of DSP applications requires the development of methodologies for the automatic implementation of floating-point algorithms in fixed-point architectures. In this paper, a new methodology for evaluating the quality of an implementation through the automatic determination of the Signal to Quantization Noise Ratio (SQNR) is under consideration. The theoretical concepts and the different phases of the methodology are explained. Then, the ability of our approach for computing the SQNR efficiently and its beneficial contribution in the process of data word-length minimization are shown through some examples.

1 Introduction

The cost and power consumption constraints of embedded systems require to use the fixed-point arithmetic for the efficient implementation of digital signal processing (DSP) algorithms. The manual transformation of floating-point data into fixed-point data is a time-consuming and error prone task. Moreover, high level development tools which allow the automation of some tasks are required for reducing the time-to-market of applications. The reduction of the development time is hindered by the manual conversion to the fixed-point level. Indeed, some experiments [4] have shown that this manual conversion can represent up to 30% of the global implementation time. Thus, methodologies for the automatic transformation of floating-point data into fixed-point data have been proposed [8, 17].

The efficient implementation of algorithms in hardware architectures (ASIC, FPGA) requires to minimize the size and the power consumption of the chip. Thus, the goal of this implementation is to minimize the word-length of the data as long as the desired precision constraints are respected. The most common used criteria for evaluating the precision of the implementation is the Signal to Quantization Noise Ratio (SQNR) [8, 6, 9]. The first stage of this implementation is the estimation of the dynamic range of the data in order to determine the word-length of their integer part. Then, the word-lengths of the data are optimized according to the desired SQNR constraint. The achievement of this second stage is based on the availability of a tool allowing the evaluation of the quality of the implementation through the determination of the SQNR at the output of the system. Most of the available methodologies are based on simulation [3, 7, 8, 14].

In this paper, a new method for the SQNR evaluation of a software or hardware implementation, based on an analytical approach is presented. This method uses a realistic noise model and allows to compute automatically the expression of the SQNR in non-recursive structures and in linear recursive structures. The use of this method in the process of data word-length minimization reduces significantly its minimization time compared to the simulation based methods.

After an overview of the available methods for SQNR evaluation, the interests of our approach in relation to the previous methodologies are underlined. In section 3, the theoretical concepts of the method are explained. Then, the techniques used for implementing this method are detailed in section 4. Finally, the ability of our approach for computing the SQNR efficiently and its beneficial contribution in the process of data word-length minimization are shown through some examples in section 5.

2 Problem presentation and related work

Two types of method can be considered for evaluating the SQNR at the output of a system. Firstly, the simulation based methods are presented. The evaluation of the SQNR can be obtained with a bit true simulation of the fixed-point algorithm. A number of this type of simulator is available with high level tools such as SPW (Cadence), CoCentric (Synopsys) [14], DSP Station (Frontier Design) or Matlab-Simulink (Mathworks) [11]. As well, `C++` classes for emulating the fixed-point mechanisms have been developed as in *SystemC* [14]. These techniques suffer from a major drawback which is the time required for the simulation [3]. It becomes a severe limitation when these methods are used in the process of data word-length optimization where multiple simulations are needed. The simulations are made on floating-point machines and the extra-code used for emulating the fixed-point mechanisms of the operations increases the execution time between one and two orders of magnitude compared to a traditional simulation with floating-point data types [7]. For obtaining an accurate estimation of the noise statistic parameters, a great number of samples must be taken for the simulation. This great number of samples combined with the increase of execution time due to the emulation of the fixed-point mechanisms, leads to long simulation time. Different techniques [3, 7, 8] have been investigated for reducing this simulation time.

Moreover, the process of data word-length optimization requires to explore the design-space of the different data word-lengths. When the simulation based methods are used, this optimization is made with an iterative process where the word-lengths of the data are progressively minimized while the SQNR is greater than a threshold [8, 13, 6]. Thus, the fixed-point algorithm is simulated again as soon as a data word-length is modified and the optimization time for a complex system becomes very huge. For reducing the number of simulations, the exploration of the design space is based on heuristic search algorithms which limit this design space [13].

An alternative to the simulation based method can be an analytical approach which determines the expression of the noise power at the output of the system according to the statistical parameters of the different noise sources. For this approach, two advantages can be underlined. Firstly, this method gives an analytic expression of the SQNR and thus provides more information about the noise behaviour in the system than a simulation based method which only gives the numerical value of the SQNR. This approach allows to analyse more precisely the influence of a particular operation and to investigate more efficiently the different possible structures of a DSP system. Secondly, the requisite execution time for evaluating the noise power is definitely lower, especially for the process of data word-length optimization in hardware design. Indeed, the determination of the SQNR expression $SQNR(b_k)$ is done only once. Then, the word-lengths of the data b_k are obtained by minimizing the size $S(b_k)$ of the chip as long as the SQNR is greater than the desired SQNR

$$\min_{b_k \in \mathcal{Z}^+} \Big(S(b_k) \Big) \quad \text{such as} \quad SQNR(b_k) \geq SQNR_{min} \qquad (1)$$

Analytical expressions of the SQNR have been formulated for some particular DSP applications as in [10]. In [15], the author has proposed a SQNR evaluation methodology based on an analytical approach. For each type of operator the output noise is modelised by the sum of the input noises propagated through the operator and the noise generated by the operator if a cast operation occurs. Different restrictive assumptions have been made for defining the expressions of the output variance of the operators. Then, the variance of the system output is obtained by traversing the signal flow graph (SFG) of the application from the inputs to the output. This method requires that the SFG is a directed acyclic graph (DAG) and consequently it can only be applied on non-recursive structures. Thus, this method suffers of two major drawbacks. The noise model is not realistic and the method is limited to non-recursive structures.

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 laws (rounding and truncation). Moreover, this method allows to compute the SQNR in non-recursive structures and in linear recursive structures. For linear systems our approach is based on the automatic computation of the transfer function of the system from its SFG representation.

3 Theoretical concepts

3.1 Noise models

The use of fixed-point arithmetic introduces an unavoidable quantization error when a signal is quantified. A common used model for the continuous-amplitude signal quantization, has been proposed by Widrow in [16] and refined in [12]. The quantization of a signal x is modeled by the sum of this signal and a random variable b. This additive noise b is an uniformly distributed white noise that is uncorrelated with the signal x and the other quantization noises. This model has been extended for modeling the computation noise in a system resulting from the elimination of some bits during a format conversion (cast operation). More especially, the roundoff error resulting from the multiplication of a constant by a discrete amplitude signal has been studied by Barnes in [1]. The author has demonstrated that the model presented above can be used if the dynamic range of the signal is sufficiently greater than the quantum step size and if the bandwidth of the input is enough large. In [2] the number of bits eliminated during a cast operation has been taken into account for expressing the first and second-order moments of the quantization noise.

3.2 Linear systems

A linear time-invariant system made up of N_e inputs $x_j(n)$ and one output[1] $y(n)$ is considered. 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 expression of the output $y(n)$ is equal to

$$y(n) = \sum_{j=0}^{N_e-1} h_j(n) * x_j(n) \tag{2}$$

The fixed-point version of this system is detailed thereafter. Let $\widehat{x_j}(n)$ be the j^{th} quantified system input and $\widehat{H_j}(z)$ be the transfer function between the output $Y(z)$ and the input $X_j(z)$ with quantified coefficients. The use of fixed-point arithmetic gives rise to an output computation error b_y which is defined as the difference between $y(n)$ and $\widehat{y}(n)$. This output computation error is due to three kinds of source of error. The noise b_{ej} results from the propagation in the system of the input quantization noise b'_{ej} associated with the input $\widehat{x_j}(n)$. When a cast operation occurs, a quantization noise b'_{gi} is generated. The output noise resulting from the propagation of b'_{gi} is called b_{gi}. Let $H_{gi}(z)$ be the transfer function between $B_{gi}(z)$ and $B'_{gi}(z)$. These two types of noise source are modelised by an uniformly distributed additive white noise as defined in the previous section. The last type of error source b_{hj} is due to the quantization of the constants. The output error b_{hj} results from the propagation of the input signal $x_j(n)$ in the subsystem whose transfer function is $\Delta H_j(z)$. This transfer function corresponds to the difference between $\widehat{H_j}(z)$ and $H_j(z)$. In order to simplify the presentation of the results of this study, each input x_j is assumed to be a white noise. Nevertheless, the same approach can be followed for any type of input signals. A representation of the noise model of the system is given in figure 1. The final expression of the output noise b_y is equal to

$$b_y = \sum_{j=0}^{N_e-1} \Delta h_j * x_j + h_j * b'_{ej} + \sum_{i=0}^{N_g-1} h_{gi} * b'_{gi} \tag{3}$$

The approaches used for determining the expression of the statistical parameters of b_{ej}, b_{gi} and b_{hj} are identical. Indeed, these noises represent the output of a linear subsystem excited by a white noise (b'_{ej}, b'_{gi}, x_j) as defined above. In order to simplify, let b_j be the output of this linear subsystem, b'_j the input and $H_j(z)$ its transfer function. Thus, the output b_j is equal to $b'_j(n) * h_j(n)$. The noise power corresponding to the second-order moment of the output noise b_j is equal to

$$E(b_j^2) = \left(\mu_{b'_j} H_j(e^{j0})\right)^2 + \sigma^2_{b'_j} \cdot \frac{1}{2\pi}\int_{-\pi}^{\pi} |H_j(e^{j\Omega})|^2 d\Omega \tag{4}$$

[1]for multiple-output system our method is repeated for each output

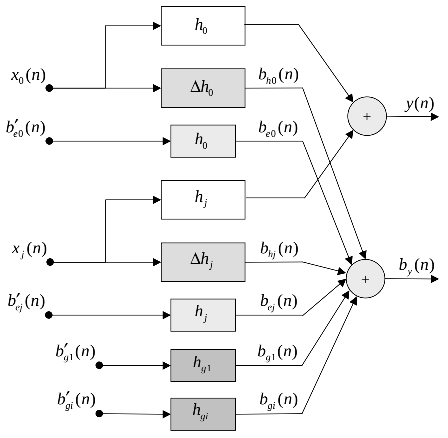

Figure 1. Noise model of the system

The output noise of the system b_y is the sum of N noises b_j corresponding to b_{ej}, b_{hj} and b_{gi}. These noises represent the output of different linear subsystems whose inputs are respectively b'_{ej}, x_j and b'_{gi}. The power of the output quantization noise is equal to

$$E(b_y^2) = \sum_{j=0}^{N-1} E(b_j^2) + \sum_{m=0}^{N-1} \sum_{\substack{l=0 \\ l\neq m}}^{N-1} \varphi_{b_m b_l}(0) \tag{5}$$

All the quantization noise sources are uncorrelated with the other noise sources and with the inputs of the system. Thus, the cross-correlation $\varphi_{b_m b_l}(0)$ is equal to the product of the means of b_m and b_l except if these two noises b_m and b_l correspond to any noise b_{hj} due to the quantization of the coefficients. In this case, the cross-correlations between the inputs of the system have to be defined. The expressions 4 and 5 show that the output noise power depends on the first and second-order moments of the quantization noise sources and the system inputs, the cross-correlation between these inputs and the frequency response of the transfer function of the different subsystems.

3.3 Non-linear systems

For non-linear systems, the concept of transfer function is no longer valid. Thus for computing the output noise, the statistical parameters of the different noise sources are propagated trough the signal flow graph (SFG) of the system. This method requires the definition of the noise propagation models for each kind of arithmetic operator. These models define the first and second-order moments of the output of an operator according to the statistical parameters of its inputs. As explained in section 2, this method can only be used in non-recursive structures.

4 SQNR computation methodology

The goal of this method is to compute the SQNR at the output of an application by using an analytical method based on the theoretical approach presented above. In this paper, the presentation of the method is restricted to the computation of the SQNR in the case of linear systems. This method uses as input, an application representation where all the fixed-point formats and parameters are specified. This representation is based on a SFG. In order to be independent of the specification language of the application, the tool is split into two parts, a front-end and a back-end. The front-end transforms the original application representation in a unique intermediate representation called G_s corresponding to the SFG of the application. At present, a front-end for interfacing the tool with our high level synthesis (HLS) tool has been developed. It computes the SFG of the application from the internal representation of the HLS tool corresponding to a data-flow graph obtained after the compilation phase. The development of a new front-end will allow to interface this method with the intermediate representation of a code generation tool for DSP.

The back-end determines the SQNR according to the analytical approach. It consists of several successive transformations (T_1 to T_3) of the SFG, described in the following sections. First of all, the SFG G_s is transformed in a graph G_{sn} representing the application at the quantization noise level. After, the transfer functions between all the inputs and the output are evaluated. Finally, the expression of the noise power is computed from the frequency response of the different transfer functions and the statistical parameters of the inputs.

4.1 Intermediate representation

The intermediate representation G_s is a signal flow graph. The nodes N_s of the oriented graph $G_s = (N_s, E_s)$ represent either an operator or a data and the edges E_s give the relation between the data and the operators. The graph G_s specifies the behaviour of the algorithm at the fixed-point level. Different information corresponding to the fixed-point data format, the type of signal and its value for the constants are associated with each data. For each operation a theoretical format representing the format of the output operator without loss of information, is defined. It will be used in the following stage for detecting every format conversion.

4.2 T_1 : Application noise model determination

The goal of the transformation T_1 is to represent the application at the quantization noise level through the graph G_{sn}. The aim of the first stage of T_1 is to detect and to include in the graph the three types of noise source defined in the section 3.2. The quantization noise parameters associated with each input of the system are defined by the user. The generated noise sources are detected by comparing the theoretical format and the real format of the data. The statistical parameters of these noises are computed from the noise model presented in [2] according to the number of bit eliminated, the format of the data and the quantization law used. The second stage of the transformation leads to the representation of the data and the operators at the noise level. Each non processed data node of G_s is split into a *signal* node and a *noise* node. Then, each operator is replaced by its noise propagation model. The models used for the multiplication and the addition are given in figure 2. The result of the transformation T_1 when a format conversion occurs, is presented in figure 3.

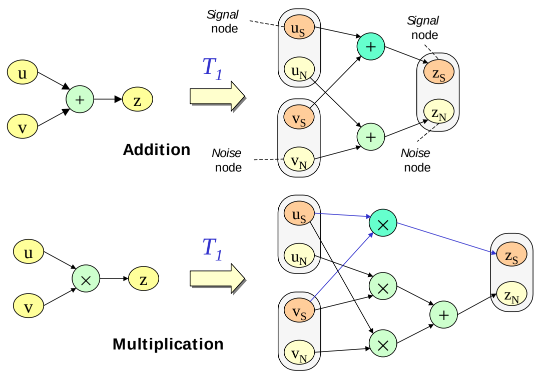

Figure 2. Operator noise models

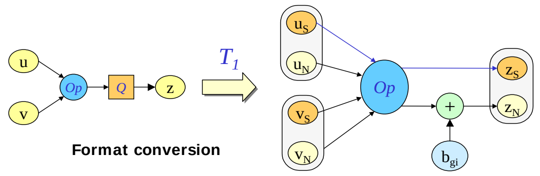

Figure 3. Format conversion

4.3 T_2 : Transfer function computation

The goal of the transformation T_2 is to determine the linear functions defining the system in order to compute the corresponding transfer functions with the $\mathcal{Z}$ transform. These linear functions 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, the use of this technique requires first of all, to transform this graph in several directed acyclic graphs (DAG). The different stages of the transformation T_2 are presented below and an example is given in figure 4.

Transformation $T_{21} : G_{sn} \rightarrow G_k$. The goal of T_{21} is to transform the graph G_{sn} into several DAG G_k if G_{sn} contains circuits. Thus, the aim of the first stage of this transformation is to detect quickly the occurrence of a circuit in the graph and to decide if the transformation T_{21} has to be applied. For minimizing the execution time of this stage, the circuit search procedure is done on the graph G_s and is stopped as early as a circuit is detected. For handling acyclic graphs with a large number of nodes, a circuit search algorithm based on a depth-first traversal of the graph is used. This algorithm allows to process each node only once.

In the second stage of T_{21}, all the circuits of the graph T_{21} are enumerated with the algorithm proposed by Johnson [5]. The dismantling of the graph in DAG requires to define the points where the circuits have to be cut. After, enumerating a circuit $C = \{a, \ldots, p1, \ldots, p_i\}$, all the paths L_i between the node a and the node representing the output are determined. The graph is dismantled at each node p_i corresponding at the last common data node between the circuit C and the i^{th} path L_i. The algorithm proposed by Johnson [5] has been modified to enumerate all the paths L_i between two nodes.

Transformation $T_{22} : G_k \rightarrow G_{eq}$. The graph $G_{eq} = (N_{eq}, E_{eq})$ is a weighted and directed graph which specifies the system with a set of linear functions. The nodes of this graph represent the output, the inputs of the system and the intermediate variables associated with the data nodes p_i. A weighted and directed edge (u, z, f_{zu}) from the node u to z indicates that the variable z is defined from the variable u with the linear function f_{zu} specified through the weight of the edge.

The goal of the transformation T_{22} is to build this graph G_{eq} from the different DAG G_k. The linear function f_z associated with a DAG G_z is obtained by a depth-first traversal of this DAG. A post-order recursive algorithm is used for traversing the DAG. At a node which is not member of the DAG sources, the algorithm examines the predecessors and then compute the linear function from the results of each predecessor and the operator. When a source is examined, the algorithm returns the name of the node i.e. the name of the data.

Transformation $T_{23} : G_{eq} \rightarrow G_{Hi}$. The graph $G_{Hi} = (N_{G_{Hi}}, E_{G_{Hi}})$ is a weighted and directed graph which specifies the algorithm with a set of intermediate transfer functions. The nodes of the graph G_{Hi} are member of the set of nodes N_{eq}. The weighted and directed edges specify the intermediate transfer functions between the head and the trail of these edges. The goal of the transformation T_{23} is

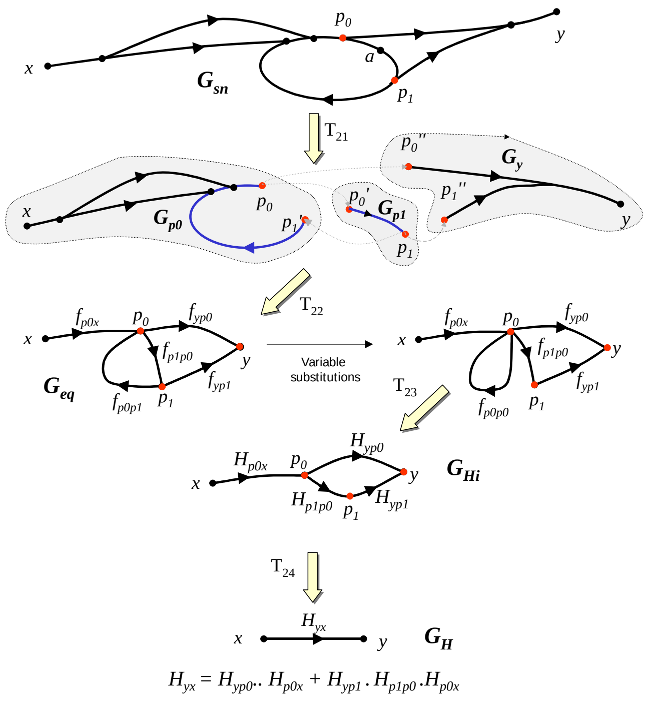

Figure 4. Transformation T_2 example

to compute these intermediate transfer functions from the linear functions.

The transfer function of a subsystem can be directly computed from the $\mathcal{Z}$ transform of the linear function defining the output y if this linear function contains explicitly all the terms related to the delayed versions of y. Thus, if any input of this subsystem is a linear function of the output y, variable substitutions must be made before the transfer function determination. Thus, after the detection of the circuits in the graph G_{eq}, a set of variable substitutions between the linear functions are made in order to eliminate these circuits (except loops) for respecting the rule presented above. Then the intermediate transfer functions are obtained by the $\mathcal{Z}$ transform of the linear functions. The variable substitution process allows to be independent of the way that the graph G_{sn} is dismantled and thus leads to the determination of the exact transfer functions.

Transformation $T_{24} : G_{H_i} \rightarrow A_H$. The graph $A_H = (N_{A_H}, E_{A_H})$ is a weighted tree which specifies the algorithm with a set of global transfer functions between the output and each input of the system. This tree represents the modelisation of the system given at the figure 1. In the graph G_{H_i}, the path between each system input and the output is determined and the global transfer function is defined as the product of the intermediate transfer functions member of this path.

4.4 T_3 : SQNR computation

The output noise power is computed from the expressions 4 and 5. 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 specified by the user or is computed from the transfer function of the system and the parameters of the system inputs according to the same method.

5 Results

5.1 SQNR computation results

The ability of our method for computing the SQNR has been successfully verified on several classical DSP applications such as FIR and IIR filters and the FFT algorithm. The estimations of the output noise power obtained with a bit true simulation and with our method are very closed. The relative error between these two estimations is included between 0.29% and 8.2% for different implementations of a second-order IIR filter and smaller than 1.5% for a 16 taps FIR filter. Two different reasons can explain the difference between these two estimations. First, the accuracy of the estimation based on simulation depends on the number of samples used. Secondly, for our method, two elements can influence the accuracy of the estimation. A slight error can be present due to the assumptions made on the linear noise model. Given that the transfer functions obtained are exact, the accuracy of the output noise power depends of the accuracy of the quantization noise model used for cast operations. Despite of the relatively high accuracy of the model proposed in [2], further improvements can be obtained by combining it with the model proposed in [1].

The execution time of the different parts of the tool has been measured. Most of the time is consumed by the transformation T_2 and especially by the circuit and path enumeration procedure (transformation T_{211}). The execution times of the transformations T_2 and T_{211} for different applications are reported in table 1. The global SQNR computation times for a fourth order cascaded IIR filter (IIR 4) and 256 taps FIR filter (FIR 256) are smaller than 1 second. These results show the efficiency of our approach on SFG with multiple cycles and on acyclic SFG with a great number of nodes. These results are smaller than those obtained with a simulation based method. Indeed, the global SQNR computation time for a second-order IIR filter simulated with the MATLAB's *Fixed-Point Toolbox* [11] is around $34s$ for 100000 input samples. In [8], the best simulation time obtained for a fourth-order IIR filter is $16.3s$.

Applications	Execution time (s)	
	T_{211}	T_2
IIR 2	0.07	0.08
IIR 4	0.43	0.45
IIR 4 (cascaded)	0.64	0.65
FIR 16	–	0.01
FIR 256	–	0.86

Table 1. Execution time

5.2 Application to data word-length optimization

This method can be integrated in a process of data word-length optimization. It can replace efficiently a SQNR evaluation method based on simulation as in [9]. The first stage of this process is the computation of the dynamic range of the data. From these results the word-length of the integer part of the data is defined in order to avoid the occurrence of an overflow. Moreover, the different format conversion operations are included in the application in order to respect the fixed-point arithmetic rules. Then, our methodology is used for evaluating the expression of the output noise power. Finally, the expression of the chip area is minimized as long as the output noise power is smaller than a threshold as defined below

$$\min_{b_k \in \mathcal{Z}^+} (S(b_k)) \text{ such as } P_{b_y}(b_k) \le P_y 10^{\frac{-SQNR_{min}}{10}} \quad (6)$$

A simplified model for the chip area evaluation has been used. Let b_{in} be the input word-length of an operator (adder or multiplier), the area of these operators is equal to

$$\begin{array}{ll} \text{adder}: & S_{add} = K_{add}.b_{in} \\ \text{multiplier}: & S_{mult} = K_{mult}.(b_{in})^2 \end{array} \quad (7)$$

A constrained non-linear minimization method has been used for solving the minimization problem given in equation 6 with $b_k \in \mathcal{R}^+$. From this solution a branch-and-bound algorithm with a limited search space is used for obtaining the solution with $b_k \in \mathcal{Z}^+$. The limitation of the search space allows to obtain a relatively small global minimization time.

A set of experiments has been performed with a second-order IIR filter whose SFG is presented in figure 5. The design space has been explored by progressively increasing the number N_v of distinct variables b_k in equation 6 and for each experiment the word-length has been optimized as explained before. The results are given in table 2 for a SQNR constraint of 80 dB. For the first experiment, all the signals have the same word-length. The second experiment provides the optimal temporal implementation where only one multiplier and one adder are used for the different operations. The last experiment corresponds to the optimal spatial implementation.

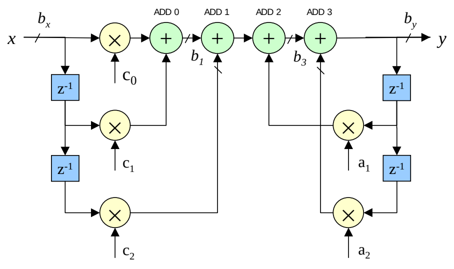

Figure 5. IIR filter signal flow graph

N_v	Word-length						$SQNR$
	b_x	b_y	b_1	b_2	b_3	b_4	(dB)
1	18	18	18	18	18	18	84.1
2	17	17	21	21	21	21	85.1
3	17	17	20	20	21	21	85.0
3	14	17	21	21	21	21	80.5
6	14	17	20	20	21	21	80.4

Table 2. Optimum word-length

6 Conclusion

A new methodology for computing the output SQNR of an application based on an analytical approach has been presented. The ability of our approach for computing the SQNR efficiently and its integration in the process of data word-length minimization have been shown through some examples.

This approach provides a significant improvement compared to the simulation based methods for most of the DSP applications. This method can not cope with the recursive non-linear systems. With our method the time required for minimizing the data word-length is definitively lower. It allows a complete design space exploration and the determination of the optimal solution.

This method will be shortly integrated in an automatic data word-length optimization methodology associated with our high level synthesis tool. Given that the execution time of an operator depends of its word-length, the phases of operator selection, scheduling, resource binding and word-length optimization have to be coupled.

The techniques presented in this paper for automatically determining the transfer functions in a signal flow graph can be used for the estimation of the dynamic range of the data in a linear system. Indeed, the evaluation of the dynamic range of a variable based on the l_1, l_2 or Chebyshev norms uses the concept of transfer function.

References

[1] C. Barnes, B. N. Tran, and S. Leung. On the Statistics of Fixed-Point Roundoff Error. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(3), 1985.

[2] G. Constantinides, P. Cheung, and W. Luk. Truncation Noise in Fixed-Point SFGs. *IEE Electronics Letters*, 35(23):2012–2014, November 1999.

[3] L. D. Coster, M. Ade, R. Lauwereins, and J. Peperstraete. Code Generation for Compiled Bit-True Simulation of DSP Applications. In *Proceedings of ISSS'98*, Taiwan, Dec. 1998.

[4] T. Grötker, E. Multhaup, and O.Mauss. Evaluation of HW/SW Tradeoffs Using Behavioral Synthesis. In *ICSPAT'96*, Boston, October 1996.

[5] D. B. Johnson. Finding All the Elementary Circuits of a Directed Graph. *SIAM Journal on Computing*, 4(1):77–84, March 1975.

[6] H. Keding, F. Hurtgen, M. Willems, and M. Coors. Transformation of Floating-Point into Fixed-Point Algorithms by Interpolation Applying a Statistical Approach. In *ICSPAT'98*, 1998.

[7] H. Keding, M. Willems, M. Coors, and H. Meyr. FRIDGE: A Fixed-Point Design And Simulation Environment. In *Design, Automation and Test in Europe 1998 (DATE-98)*, 1998.

[8] S. Kim, K. Kum, and S. Wonyong. Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs. *IEEE Transactions on Circuits and Systems II*, 45(11), November 1998.

[9] K. Kum and W. Sung. Word-Length Optimization For High Level Synthesis of Digital Signal Processing Systems. In *SiPS'98*, pages 142–151, Boston, October 1998.

[10] B. Liu. Effect of Finite Word Length on the Accuracy of Digital Filters - A Review. *IEEE Transaction on Circuit Theory*, 18(6), November 1971.

[11] Mathworks. *Fixed-Point Blockset User's Guide (ver. 2.0)*, 1999.

[12] A. Sripad and D. L. Snyder. A Necessary and Sufficient Condition for Quantization Error to be Uniform and White. *IEEE Trans. ASSP.*, 25(5):442–448, Oct. 1977.

[13] W. Sung and K. Kum. Simulation-Based Word-Length Optimization Method for Fixed-Point Digital Signal Processing Systems. *IEEE Transactions on Signal Processing*, 43(12), Dec. 1995.

[14] Synopsys. *Converting ANSI-C into Fixed-Point using CoCentric Fixe-Point Designer*. Synopsys Inc., April 2000.

[15] J. Toureilles, C. Nouet, and E. Martin. A Study on Discrete wavelet transform implementation for a high level synthesis tool. In *EUSIPCO'98*, Rhodes, Greece, Septembre 1998.

[16] B. Widrow. Statistical Analysis of Amplitude Quantized Sampled-Data Systems. *Trans. AIEE, Part. II:Applications and Industry*, 79:555–568, 1960.

[17] M. Willems, V. Bursgens, and H. Meyr. FRIDGE: Floating-Point Programming of Fixed-Point Digital Signal Processors. In *ICSPAT'97*, 1997.