Quality Evaluation in Fixed-point Systems with Selective Simulation

Riham Nehmeh

Thèse

UNIVERSITE
BRETAGNE
LOIRE

THESE INSA Rennes
sous le sceau de l'Université Bretagne Loire
pour obtenir le titre de
DOCTEUR DE L'INSA RENNES
Spécialité : Traitement du signal et de l'image

présentée par

Riham NEHMEH

ECOLE DOCTORALE : *MATISSE*
LABORATOIRE : *IETR*

Quality Evaluation in Fixed-point Systems with Selective Simulation

Thierry Michel
Ingénieur de recherche ST Microlectronics Crolles / Membre Invité

Thèse soutenue le 13.06.2017
devant le jury composé de :

Eric Martin
Professeur à l'Université de Bretagne Sud / Président
Michel Paindavoine
Professeur à l'Université de Bourgogne / Rapporteur
Lionel Lacassagne
Professeur à l'Université de Paris VI / Rapporteur
Fabienne Nouvel
Maitre de conférences HDR à l'INSA Rennes / Examinatrice
Andreï Banciu
Ingénieur de recherche ST Microlectronics Crolles / Co-encadrant
Daniel Ménard
Professeur à l'INSA de Rennes / Directeur de thèse

Quality Evaluation in Fixed-point Systems with Selective Simulation

Riham Nehmeh

En partenariat avec

Contents

List of Figures

List of Tables

CHAPTER 1 Introduction

1.1 Scope of the Thesis

In the last decades, the implementation of Digital Signal Processing (*DSP*) applications has been facilitated by the massive advance in semiconductor technology. Using digital signal processing applications allowed the emergence of new products with complex algorithms to meet the users' demands. The complexity of such applications puts the companies in a challenging cycle of design and product delivery. Companies have to deliver more value for every unit cost they charge from consumers due to competition between them. Practically, designers work in an increasing complexity environment to reduce the time-to-market and minimize the product cost while satisfying the customer demands.

Telecommunication belong to one of the fast growing industries. For example, the number of mobile networks subscribers (users) passed the staggering number of 6 billion worldwide [7]. This advance in the telecommunication industry has drawn benefits from the growing semiconductor technology. As an example, smart phones nowadays include a big combination of telecommunication, signal processing and other applications. The implementation of all these applications should comply with the demand of users in terms of quality under strict constraints related to energy, cost and time.

The cost of modern electronic devices is usually measured in terms of silicon area, power profile and execution time, which should be kept to a minimum without compromising the system performance. These goals are conflicting in nature and the designer has to inevitably derive a trade-off between the system performance and its cost. Therefore, it is very important to make careful decisions in every design step to ensure the best possible performance of the entire system. One of the first and major decisions is the choice of the arithmetic operators used to implement the algorithms, which has a large impact on the cost-performance trade-off. Floating-point and fixed-point operators are two popular choices available for the implementation of all arithmetic operations. Fixed-point arithmetic operators are known to take significantly lesser area, shorter latency and to consume less power. Implementation of telecommunication algorithms usually have rigorous performance parameters to be achieved and demand high computational power. In such cases, using fixed-point arithmetic allows satisfying such constraints thanks to its ability in manipulating data with lower word-length compared to floating-point arithmetic. Thus, digital signal processing algorithms are implemented into fixed-point architectures and floating-point to

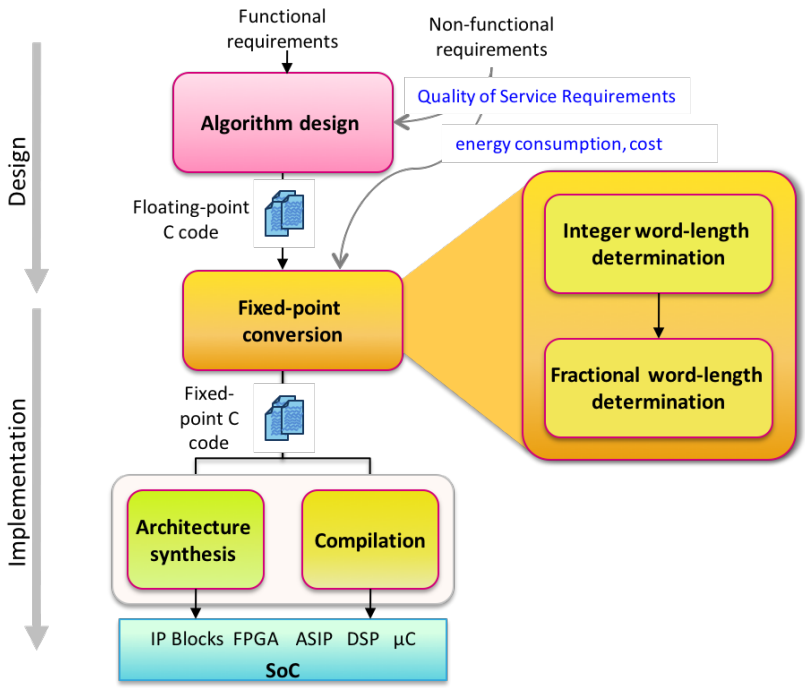


Figure 1.1 – DSP System Design Flow

fixed-point conversion is mandatory.

The various stages of the design flow of a digital signal processing application are described in Figure 1.1. The first step of this cycle is the definition of the requirements by specifying the functions to be performed by the system as well as the architectural constraints of the system (time, memory size, energy consumption). The next step is the design and the specification of a complete description of the algorithm. At this stage, simulations are carried out to ensure that the system performance is satisfied. These simulations can be performed using tools used to simulate DSP algorithms such as Matlab/Simulink (Mathworks) [8], Scilab (Inria) [9], ... Floating-point arithmetic is used to overcome problems of computation accuracy. Once the algorithm is designed, it is implemented into an embedded system using a programming language. The architectural solutions used for the implantation can be software or hardware such as ASIC, FPGA or DSP. Any practical DSP implementation uses fixed-point arithmetic to reduce the area and power consumption and obtain a cost-effective hardware. A conversion process from the floating-point description of the algorithm to a fixed-point implementation that customizes every wordlength in the datapath has to be realized. This task is the focus of this thesis.

The floating-point to fixed-point conversion process is an optimization problem [10] which derives the data word-length. This process explores the trade-off between the cost and the application quality degradation due to the limited bit-width of fixed-point data types. Moreover, it has been shown that the conversion process can take up to 30% of the total development time [11, 12]. Thus, an automatic floating-point to fixed-point conversion tool is essential to efficiently optimize

the implementation cost while satisfying the performance constraints in the shortest possible time. Recently, High Level Synthesis tools [13, 14] have emerged. These tools generate register transfer level (RTL) implementations directly from a C/C++ fixed-point specification of the application. These tools reduce significantly the development time while allowing a good design space exploration. Thus fixed-point conversion becomes the bottleneck for fast product development.

The conversion process consists of two parts corresponding to the determination of the integer part word-length and the fractional part word-length. To reduce the complexity of the conversion, the determination of the integer part and the fractional part word-lengths are handled separately. The integer word-length (*IWL*) optimization is based on determining the data dynamic range while fractional word-length (*FWL*) optimization consists on the numerical accuracy analysis. The limited integer and fractional bit-width of the fixed-point data types will introduce an overflow occurrence and quantization error respectively which generate a degradation of the application quality.

Range estimation and overflow error The dynamic range estimation determines the minimum and maximum values and is used to compute the minimum number of bits for the integer part. Classical range estimation methods compute theoretical absolute bounds that will never be exceeded in practice to avoid the appearance of overflows. In doing so, the provided ranges are pessimistic and the implementation cost is largely increased. As the absence of overflows is guaranteed, the optimization of the integer part word-length under performance constraints becomes impossible and the trade-off accuracy-implementation cost is considered only for the fractional part.

In many applications occasional overflows are acceptable if the probability of occurrence is small enough and the range estimation method should be able to take this property into account. Moreover, methods like interval and affine arithmetic do not provide additional information about the signal variation inside the range making it a poor approximation of the real variation. Signals that have large variations but have small probabilities for their extreme values are not well considered.

Different techniques have been proposed to determine the probability density function of any type of data. They allow determining the dynamic range for a given overflow occurrence probability. These techniques are based on Extreme Values Theory [15, 16, 17] or stochastic approaches like Karhunen-Loeve expansion [18, 19] and Polynomial Chaos Expansion [20].

Stochastic techniques determine the PDF of the system inputs to evaluate the data dynamic range. In linear time-invariant systems (*LTI*) the Karhunen-Loeve Expansion (*KLE*) processes a stochastic discretization of the input in terms of random variables. Using the superposition property of LTI systems, this technique determines the corresponding description of the output which take into account the temporal and spatial correlation and thus provides tighter bounds. The Polynomial Chaos Expansion (*PCE*) is used for non LTI systems. Compared to KLE, a wider range of systems are supported but at the detriment of a higher computational complexity. Extreme Values Theory is used with lightweight simulations to obtain samples of the output that can further be used to estimate the probability distribution.

In general, these techniques link the overflow probability and the dynamic range. However,

defining a mathematical expression of the application quality metric is not trivial and complicates the automation of fixed-point system design. On the one hand, each application has its own quality metric. For example, the bit error rate is used for communication system and the mean opinion score can be used for audio applications. On the other hand, a direct correlation between the application quality metric and the overflow probability is hardly established.

Numerical accuracy analysis and un-smooth error The numerical accuracy analysis is linked with the notion of quantization noise. This analysis studies the sensitivity of the output to slight changes at the input by computing an error metric which can be for example the signal-to-quantization-noise-ratio (SQNR). In fact, several works are focused on optimizing the fractional part word-length using the quantization noise power as an error metric.

The quantization process can be modeled as a random process with specific characteristics. At system-level, the single noise source (*SNS*) [21] can be used to model the total quantization noise at the output of a system as a sum of various random processes. The characteristics of the random process are determined from the knowledge of the quantization noise power and its spectral and distribution functions.

The models describing the quantization noise based on Widrow's quantization models [22] and perturbation theory are accurate only when the quantization step size is very small in comparison to the dynamic range. As the quantization step-size increases, the noise properties deviate from the analytical predictions and soon become intractable. Such quantizers are referred to as un-smooth quantizers.

Moreover, existing methods of numerical accuracy analysis evaluate only the power of the output quantization noise. In some cases, like the evaluation of the performance in systems with un-smooth operators [5], this limited information is insufficient and the entire probability density function of the noise should be determined.

As a solution, simulation based approaches are used to overcome the limitation of classical approaches. In comparison to stochastic approaches, where establishing the link between the application quality metric and error occurrence probability is not trivial in the general case, simulation based approaches can be performed on any type of system [8, 23, 24]. However, they are time consuming and require a large number of samples to obtain an accurate analysis. This results in a serious limitation on the applicability of simulation based methods especially that the existing simulation (complete simulation) based approaches simulate all the input samples in every evaluation of the quality analysis.

1.2 Objectives and Contributions of the Thesis

In this thesis, we aim at accelerating the process of floating-point to fixed-point conversion. We present our new approach based on a framework using *selective simulations* to accelerate the simulation of overflow and un-smooth error effects. This approach can be applied on any C/C++ DSP application to evaluate the degradation due to overflow or un-smooth errors. Compared to

complete fixed-point simulation based approaches, where all the input samples are processed, the proposed approach simulates the application only when an error occurs. Indeed, overflows and un-smooth errors must be rare events to maintain the system functionality. Consequently, selective simulation allows reducing significantly the time required to evaluate the application quality metric.

As most of the existing works is focused on optimizing the fractional part, we focus on optimizing the integer part, which can significantly decrease the implementation cost when a slight degradation of the application performance is acceptable. Indeed, many applications are tolerant to overflows if the probability of overflow occurrence is low enough. Thus, we exploit the proposed framework using a new integer part word-length optimization algorithm to accelerate the optimization of the IWL. Furthermore, we apply the framework of selective simulation to evaluate the accuracy in fractional word-length optimization of system with un-smooth operators.

The different contributions of this thesis and the associated publications in international conferences and journals are listed below:

- The design and the development of a selective simulation framework to avoid complete fixed-point simulations to evaluate the application quality metric.
  - The exploitation of the selective simulation framework to accelerate the analysis of overflow effect. This contribution has been published in the DASIP conference [25] and in the Journal of Signal Processing Systems [26]
  - The exploitation of the selective simulation framework to accelerate the analysis of un-smooth error effect.
- The design and the development of an optimization algorithm for the integer part word-length. This contribution has been published in ICASSP conference [27] and in the Journal of Signal Processing Systems [26]

1.3 Outline of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 presents the fixed-point and floating-point arithmetic, and compares them in terms of arithmetic aspects as well as software and hardware implementation. It also describes the steps of the floating-point to fixed-point conversion and explains the existing methods for both the range estimation and the accuracy analysis. Furthermore, this chapter provides a description of ID.Fix tool. This tool allows the conversion of a floating-point C source code into a C code using fixed-point data types.

Chapter 3 explains the problem of evaluating the system when an error (overflow or un-smooth error) occurs with low probability. It explains the concept of overflow and its relation with integer part word-length, and the concept of un-smooth error and its relation with the quantization process. Moreover, a new approach that uses selective simulations to accelerate the simulation of finite precision effects analysis is presented. This approach can be applied on any $C/C++$-based DSP application using fixed-point arithmetic to evaluate the degradation due to finite word-length.

As a next step, Chapter 4 details the proposed selective simulation approach and explains its

procedure when applied in the presence of overflow. After presenting the methodology of variables selection for integer word-length optimization, the implementation of the proposed framework in $C++$ is explained. Then, experiments are conducted to verify the effectiveness of the proposed selective simulation technique.

In Chapter 5, the proposed selective simulation approach is exploited to design a new optimization algorithm for integer part word-length optimization. The fixed-point refinement explores the trade-off between implementation cost and application quality. The cost and the application quality of systems using fixed-point arithmetic depend on the word-length of each data. After presenting the existing algorithms for determining the word-length, the proposed optimization algorithm is explained.

Chapter 6 considers the use of selective-simulation technique to evaluate the effect of un-smooth errors. The detection of potential un-smooth errors is detailed and then, experiments on an image processing application are conducted to evaluate the efficiency of the proposed technique.

Finally, in Chapter 7, we conclude this thesis and present several potential research directions to be developed in future work.

CHAPTER 2 State Of The Art

Most digital signal processing algorithms manipulate data representing real numbers. The design of signal processing algorithms rather uses floating-point arithmetic because of its development simplicity and its good numerical properties. However, for numerous embedded systems, fixed-point arithmetic is preferred because of its higher benefit in terms of power consumption, area and architecture latency. Thus, a conversion from floating-point to fixed-point is required before the hardware implementation of the algorithm. This chapter first presents and compares fixed-point and floating-point arithmetics. Then, the steps of the conversion process and the existing fixed-point conversion methods are detailed. Especially, existing techniques for dynamic range and accuracy evaluation are presented. Finally, the fixed-point design environment ID.Fix is described.

2.1 Fixed-Point representation

2.1.1 Number representation

In almost all digital platforms, digital systems use binary number representation and specify the corresponding rules for performing arithmetic operations (addition, multiplication, etc.). The exact representation of numbers requires infinite precision. However, only a limited number of bits can be used in practice. Thus, most of the time, scientific calculations give approximations of the exact values.

In general, digital signal processing algorithms require high computing capabilities. Therefore, choosing the right number representation format is critical in the implementation of any digital signal processing algorithm. Fixed-point and floating-point arithmetics are generally used for storage and computation purpose.

The effect of limited number of bits for standard arithmetic can be evaluated from two different perspectives. The first one is concerned with the numerical accuracy, which is determined by the quantization step of the system. The second aspect is related to the variation of the maximum dynamic range allowed by the representation. The dynamic range is the domain of possible values that can be represented for the considered format. It is evaluated as in Equation 2.1, where X_{MAX} and X_{MIN} are respectively the largest and the smallest magnitude that can be represented by the coding standard.

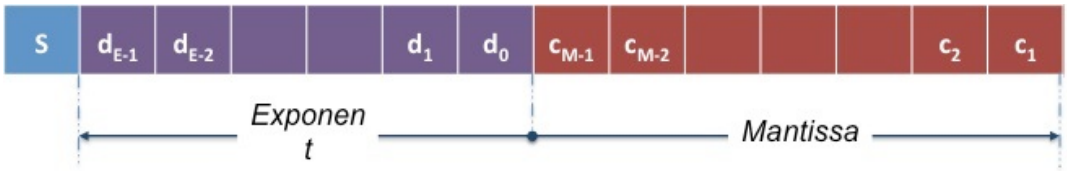


Figure 2.1 – Floating-point representation of a signed-number

$$D_{dB} = 20\log_{10}\left(\frac{X_{MAX}}{X_{MIN}}\right) \tag{2.1}$$

In the next sections, floating-point and fixed-point representations are described.

2.1.2 Floating-point representation

The floating-point arithmetic is widely used when high precision is required. It represents a real number composed of a sign s (+ or -), a scale factor (exponent) and a fractional part (mantissa) as shown in Figure 2.1. The mantissa determines the precision of the represented number and the exponent defines the power of the base (typically 2 or 10). The latter specifies the position of the binary point and it is used as an explicit scale factor that changes during computations to code accurately small and high values.

Let x be a real signed number represented in floating-point with a base B, a sign S, a mantissa M and an exponent E. The value associated with x in the floating-point representation is given by the following expression

$$x = (-1)^S M B^E \tag{2.2}$$

Given that several couples of mantissa and exponent can be used to represent a same value, a standard floating-point format has been introduced to remove this ambiguities. Today, the IEEE standard for binary floating -point arithmetic (IEEE 745-2008) is used in most CPUs to ensure the portability of computing software. It specifies the floating-point format and the rounding modes. It describes not only how the arithmetic operations must be performed, but also the treatment of possible exceptions (division by zero, overflow, ...). The mantissa and the exponent are encoded with sign and absolute value representation. The numbers are normalized to ensure a unique encoding. The mantissa is normalized to represent a value in the interval $[1, 2[$. As a result, the value of its most significant bit is implicitly equal to 1 and is not represented. The exponent is encoded as an unsigned number, so it is coded relatively to a bias $b = 2^{E-1} - 1$ in order to represent numbers smaller than 1. The bias depends on the number of bits that are allocated for the representation of the exponent. It is equal to 127 for 32-bit floating-point (single precision) and 1023 for 64-bit floating-point (double precision). The value of the data x represented in the binary IEEE 754 floating-point format is

Representation type (bits)	Sign	E	M	Bias
Half Precision	1 [15]	5 [14-10]	10 [9-0]	15
Single Precision	1 [31]	8 [30-23]	23 [22-0]	127
Double Precision	1 [63]	11 [62-52]	52 [51-0]	1023

Table 2.1 – Bit allocation in the binary IEEE 754 floating-point format coding. Numbers in brackets represent the bit positions.

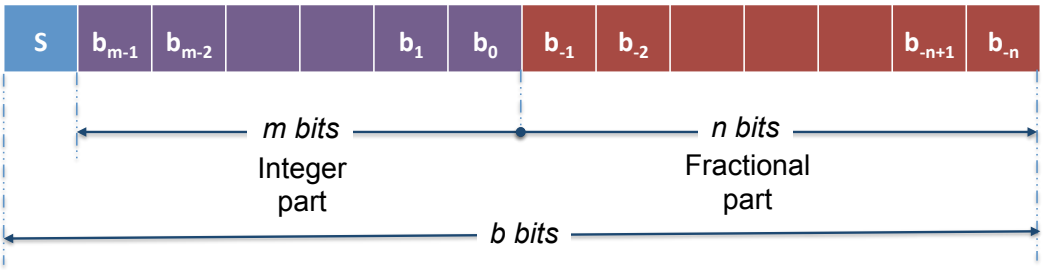


Figure 2.2 – Fixed-point representation of a signed-number

$$x = (-1)^S * 2^{E-b} * (1 + M) \tag{2.3}$$

The bias b is introduced to represent numbers smaller than 1. The mantissa M and the exponent E are calculated as follows

$$M = \sum_{i=1}^{m} m_i 2^{-i} \tag{2.4}$$

$$E = \sum_{i=0}^{e-1} e_i 2^i \tag{2.5}$$

The standard defines the single-precision data type on 32-bits and the double precision data type on 64-bits. More recently, the half-precision format on 16-bits has been proposed. The table 2.1 shows the distribution of bits of the different types represented by the IEEE 754 norm. The terms WL, M and E represent, respectively, word-length, mantissa and exponent. The standard also reserves some values to represent the values 0, $-\infty$ and $+\infty$ and NaN (Not a Number).

The exponent range for normalized numbers is $[-126, 127]$ for single precision format (32 bits) and is $[-1022, 1023]$ for double precision format (64 bits) and $[-14, 15]$ for half precision format (16 bits).

2.1.3 Fixed-point representation

The representation of fixed-point format is composed of an integer part and fractional part. The binary point in fixed-point representation and the number of bits of each part are fixed. Thus, the scale factor of the associated data is constant and the range of the values that can be represented does not change during the computation.

Figure 2.2 presents the general representation of a number in fixed-point format composed of a sign bit S (the most significant bit) and $b-1$ bits divided between the integer and the fractional parts. m and n represent the position of the radix point respectively to the most significant bit (*MSB*) and to the least significant bit (*LSB*). They correspond to the number of bits respectively for the integer part and for the fractional part if they are positive. The bit sign S is equal to 1 if the represented value is negative value and to 0 if it is positive. Let $Q_{m.n}$ be the fixed-point format with m bits for the integer part and n bits for the fractional part. In this representation, each bit of the integer and fractional parts corresponds to a power of 2. Intuitively, the bits of the integer part provide the representation of the number with positive powers and the bits of the fractional part provide the representation of the number with negative powers of 2 (2^{-1}, 2^{-2},..., 2^{-n}).

It should be noted that the sign bit can be removed if the numbers are always positive. Moreover, some parts of the representation can removed leading to a negative value for parameters m or n. A negative value for m means that the values have an amplitude significantly lower than 1. The first few bits of the fractional part (power 2^{-1}, 2^{-2}, ...) have not to be specified since their values do not vary. Similarly the term n can be negative to represent data for which the step-size between two consecutive values is higher than 1.

In general, fixed-point numbers are encoded using two's complement standard. This standard has some interesting arithmetical properties regarding addition and subtraction operations. It also allows a unique representation of 0. Thus, the domain of possible values is not symmetric with respect to the origin, having $2^{(m+n)}$ strictly negative values and $2^{(m+n)}-1$ strictly positive ones. The value of a fixed-point number x using two's complement representation is expressed as:

$$x = -2^m S + \sum_{i=-n}^{m-1} b_i \cdot 2^i \tag{2.6}$$

The maximal and minimal values that can be represented depend on the binary point location and the number of bits used in the fixed-point representation. The definition domain corresponding to the values that can be represented by the format $Q_{m.n}$ is given in Equation 2.7. Moreover, the quantization in such arithmetic is uniform and is constant for the entire dynamic scale: $q = 2^{-n}$.

$$D = [-2^m; 2^m - 2^{-n}] \tag{2.7}$$

Thus, the finite word-length in the case of fixed-point representation corresponds to a trade-off between the dynamic range and the precision. On the one hand, increasing the integer part word-length will result in the extension of the dynamic range. On the other hand, increasing the fractional word-length part will decrease the quantization step resulting in the enhancement of the

accuracy.

2.1.4 Comparison between fixed-point and floating-point arithmetic

In this section, a comparison between the floating-point and the fixed-point arithmetic is presented. This comparison includes both numerical quality and hardware implementation.

2.1.4.1 Numerical quality

At the arithmetic level, the quality of fixed-point and floating-point representation are compared by analysing the dynamic range and the Signal to Quantization Noise Ratio (SQNR).

Dynamic range analysis One of the most important criteria to compare different arithmetic is the dynamic range. The dynamic range ratio D is defined as the ratio between the largest magnitude value X_{MAX} and the smallest magnitude value X_{MIN} of the signal that can be represented excluding the value 0. It is expressed in decibels as defined in Equation 2.1.

The dynamic range of the floating-point representation having E bits for the exponent can be determined as in Equation 2.8.

$$D_{dB} \simeq 20\log_{10}\left(2^{2K+1}\right) \quad \text{with} \quad K = 2^{E-1} - 1 \tag{2.8}$$

For the fixed-point representation, the dynamic range is linear with the number of bits b used in the representation. It is expressed as

$$D_{dB} = 20\log_{10}(2^{b-1}) = 20(b-1)\log_{10}(2) \tag{2.9}$$

The coding of a number in fixed-point representation is done by mapping the real value of x by another value $\hat{x}$ from the coding domain. When the number exceeds the allowed range of values defined by the coding standard, *i.e.* $x \notin [\hat{x}_{min}, \hat{x}_{max}]$, an overflow occurs.

Figure 2.3 shows the evolution of the dynamic range with respect to the number of bits used in the fixed-point and the floating-point representation. For floating-point, it is considered that the number of bits allocated to the exponent is $E = [\frac{b}{4}]$. The floating-point arithmetic leads to an exponential evolution of the dynamic range depending on the number of bits. The fixed-point arithmetic leads to a linear evolution of the dynamic range depending on the number of bits. The dynamic range obtained with a fixed-point representation is greater than the dynamic range of a floating-point representation for number of bits less than 16. This trend is reversed when the number of bits exceeds 16. For 32-bits data (single precision floating-point data type), floating-point representation shows a big interest as it provides higher dynamic range compared to fixed-point representation.

As an example, Table 2.2 presents the dynamic ranges of single precision floating-point representation and some fixed-point representations.

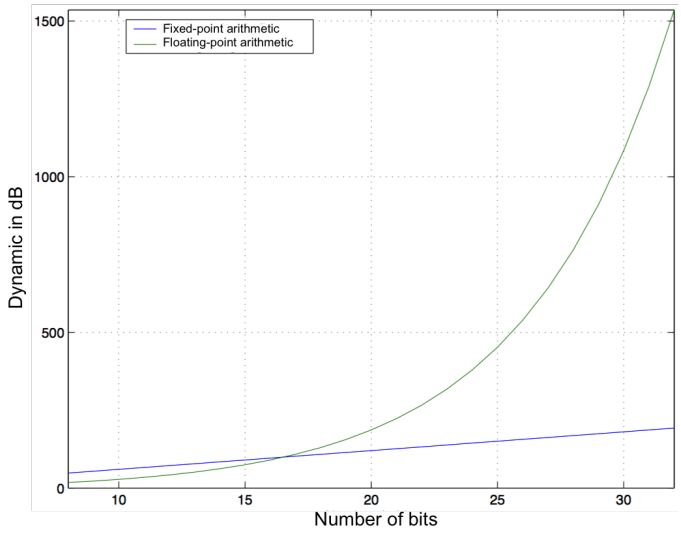


Figure 2.3 – Dynamic range evolution depending on the total number of bits [1]

Representation	Dynamic range (dB)
Single precision floating-point (32 bits)	1535
Fixed-point (16 bits)	90.3
Fixed-point (32 bits)	186.63
Fixed-point (64 bits)	379.29
Fixed-point (128 bits)	764.61

Table 2.2 – Dynamic range for different fixed-point and floating-point formats

Quantization noise analysis Another important criterion to compare arithmetic is the numerical precision. It represents the step between two successive numbers and depends on the quantization process.

For floating-point number representation, the precision is proportional to the magnitude of the encoded number. As the number increases, the value of the exponent needed to encode it increases. Consequently, the quantization step, *i.e* the distance between two consecutive numbers, increases.

The quantization step q is bounded as follows

$$2^{-(m+1)} < \frac{q}{|x|} < 2^{-m} \tag{2.10}$$

where m is the size of the mantissa and $|x|$ is the magnitude of the represented number. This relation shows how the quantization step is adapted to the magnitude of the number. When the magnitude of the number is small, the quantization step is also small and the precision is still good. Similarly, the precision decreases when the magnitude of the number increases. A detailed analysis of the floating-point quantization noise can be found in [28, 29].

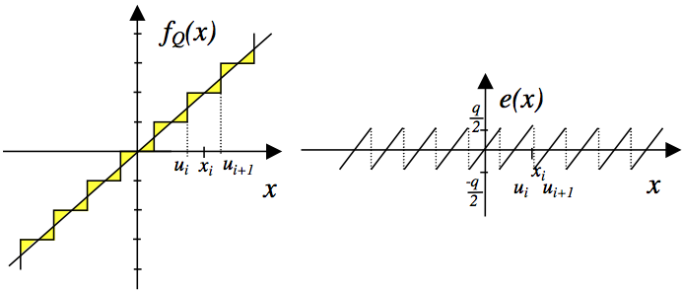


Figure 2.4 – Quantization process with rounding

For fixed-point numbers, the quantization process generates a loss of precision as only a finite numbers of possible values can be represented. More specifically, 2^b distinct values can be represented if b bits are used. However, the precision of the system depends on the binary point position and the number of bits for the fractional part. Given that q is the difference between two consecutive numbers (quantization step), the value of q depends on the LSB weight 2^{-n}. The quantization noise is the difference between the real value x and the fixed-point representation $\hat{x}$ (Equation 2.11).

$$e(x) = \hat{x} - x \tag{2.11}$$

Several quantization modes are available such as the round-off and the truncation modes. They are presented respectively in Figures 2.4 and 2.5 where $f_Q(x) = \hat{x}$. When considering the round-off mode, the number is coded to the nearest quantization level as presented in Equation 2.12. In this case, the maximum quantization error is $\pm\frac{1}{2}LSB$. $e(x)$ has a null mean and belongs to $[-\frac{q}{2}, \frac{q}{2}]$.

$$\hat{x} = u_i + \frac{q}{2}, \quad \forall x \in \Delta_i = [u_i, u_{i+1}[\tag{2.12}$$

The truncation mode chooses the inferior quantization level for the representation of a number as in Equation 2.13. Thus, the maximum quantization error of this mode is q and $e(x) \in [0, q]$.

$$\hat{x} = u_i, \quad \forall x \in \Delta_i \tag{2.13}$$

Signal-to-Quantization Noise Ratio The signal-to-quantization noise ratio (*SQNR*) is the most common criterion used in DSP applications to describe the computational accuracy. The SQNR is defined as the ratio between the power of the represented signal P_x and the power of the quantization error P_e. It is expressed in dB as

$$SQNR_{dB} = 10 \log \left(\frac{P_x}{P_e} \right) \tag{2.14}$$

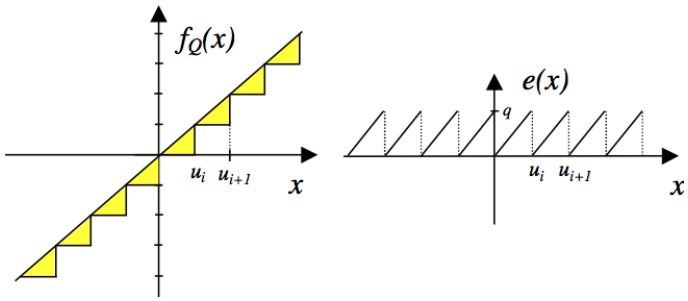


Figure 2.5 – Quantization process with truncation

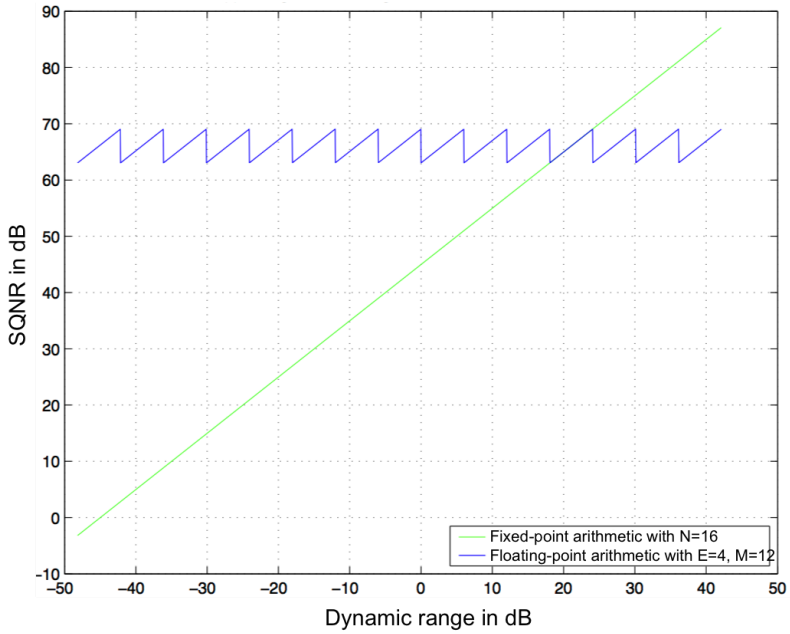


Figure 2.6 – SQNR evolution depending on the dynamic range of input signal [1]

Let D_x be the dynamic range of the signal x, and K_x the ratio equal to $\frac{\sqrt{P_x}}{D_x}$. P_x can be expressed as

$$P_x = \lim_{b\to\infty} \frac{1}{b} \sum_{k=0}^{b-1} x(k)^2 = (K_x D_x)^2 \tag{2.15}$$

In the case of fixed-point arithmetic, the SQNR is defined by Equation 2.16. The equation shows that the SQNR is linearly dependent on the signal amplitude.

$$SQNR_{dB} = 20\log(D_x) + 20\log(K_x) - 10log(P_e) \tag{2.16}$$

In contrast to the floating-point representation, the fixed-point representation has a quantization step that is proportional to the amplitude of the signal. The expression of the SQNR for round-off

quantization mode is presented in [30]. It depends on the number of bits used for the representation of the mantissa. Figure 2.6 illustrates the evolution of the SQNR with respect to the dynamic range of a 16-bits data for both floating-point and fixed-point arithmetic. In the case of floating-point representation the SQNR is almost constant. That is due to the use of an explicit exponent which allows the adjustment of the SQNR. In the case of fixed-point representation the SQNR is proportional to the dynamic range. For low dynamic range, the signal in fixed-point representation is very sensitive to quantization error and the floating-point representation provides a better SQNR. However, the quantization step of the floating-point representation becomes higher than the fixed-point representation. Thus, for this case of 16-bits, the fixed-point representation can guarantee a SQNR higher than the one obtained for floating-point representation, if it is properly scaled. We should also note that the choice of the number of bits allocated to the mantissa and to the exponent in the floating-point representation is a trade-off between a high dynamic range and a high SQNR.

2.1.4.2 Comparison of software and hardware implementation

Each software or hardware implementation has its own constraints that determine the choice of the most suitable arithmetic. In the case of a software implementation, the data word-lengths are fixed and a limited number of word-length are available in both fixed-point and floating-point representation. In typical software platforms, the sizes of the integer data types that can embed fixed-point data are usually a byte (8 bits), half of a word (16 bits), a word (32 bits) and a long as double word (64 bits). The floating-point numbers are usually in two formats: either single or double precision accuracy. The half-precision format is supported by some processors. In addition, it should be noted that some processors support operations on vectorized data with the SIMD (Single Instruction Multiple Data) concept [31]. Such instructions operate on a set of data of same size and type collected in a data block of fixed size, which is called a vector. Then, it is possible to reconfigure the data path according to a standard size (the number of bits is usually a multiple of four or eight) in order to control the data word-length in fixed-point arithmetic.

In hardware implementation, any word-length is supported for fixed-point arithmetic. Moreover, hardware implementation opens horizons for custom floating-point units [32], [33]. In [34], specifications of floating-point operators are described using a C++ library of floating-point operators. This results in automating optimal implementations of floating-point operators in the hardware implementation so that the computing time is small enough to achieve the desired operating frequency. The impact of the number of bits allocated to the floating-point operator in terms of dynamic range and precision is not as simple as in the case of fixed-point representtaion. Ultimately, it is difficult to make a choice between fixed-point format and floating-point format without explicitly exploring all the operations.

The floating-point arithmetic has the advantage of having a greater dynamic range for data with more than 16-bits and a better SQNR compared to fixed-point arithmetic. Nevertheless, as the exponent varies for different data, the floating-point arithmetic operations are complex to perform. For example, the floating-point addition is performed in three steps. First, the two input data are denormalized in order to have a common format. Then, addition is performed. Finally, normalization is carried-out to adjust the exponent of the addition output. Thus the cost

of a floating-point addition is high compared to the cost of a fixed-point addition. In this case, the binary position of the two input operands are aligned by shifting the bits and then the two operands are added.

Low power consumption is one of the major requirements of embedded systems. Floating-point arithmetic based on the IEEE-754 standard requires the use of at least 32 bits for data (half-precision is supported by few processors). However, the majority of applications based on fixed-point arithmetic use data with limited word-length: from 6 to 16 bits for input data and less than 32 bits for intermediate results. Thus, the widths of buses and memories in the fixed-point architectures are lower. This leads to lower energy consumption and cost for architectures based on fixed-point arithmetic, in addition to the fact that operators in fixed-point are less complex. For example, performing 32-bit fixed-point addition and 16-bit fixed-point multiplication require $0.5pJ$ and $2.2pJ$ respectively, while a 64-bit floating-point unit consumes $50pJ$ [35]. Therefore, the fixed-point arithmetic is preferred for algorithms implementation in embedded systems. However, fixed-point arithmetic has the disadvantage of having a lower SQNR, and therefore a lower numerical accuracy. Thus, it is necessary to pay attention to the numerical accuracy in fixed-point implementation. The accuracy can be determined either by simulations or by analytical methods as presented in Section 2.2.2.

in summary, the operations used in embedded applications with fixed-point arithmetic are less expensive compared to delivering data and instructions to the functional unit of a programmable floating-point system. However, the limited bit-width results in a degradation of the numerical accuracy. In addition, using fixed-point format causes overflow occurrence whenever the integer word-length (*IWL*) is insufficient to represent the dynamic range variation. Thus, DSP algorithms are implemented into fixed-point architectures and floating-point to fixed-point conversion is mandatory in order to find the compromise between area and arithmetic precision.

2.2 Floating-point to fixed-point conversion

The conversion of floating-point representation into a fixed-point representation or the refinement of an existing fixed-point specification is an optimization process divided into two main steps. The first step corresponds to the determination of the integer word-length of each variables. After the evaluation of the definition domain of each data, where the width (number of bits) must represent all the possible data values, the position of the binary point is determined while minimizing the integer part of each data. The second step is the fractional word-length determination, which defines the numerical accuracy. This section presents the existing techniques for evaluating the dynamic range and the numerical accuracy in order to determine the word-length of each variable.

2.2.1 Integer word-length optimization

The integer world length determination is a critical step in the fixed-point conversion process. The number of bits for the integer part of a data x depends on its dynamic range and is linked with the probability density function $f(x)$ the data x. If the extreme values (maximum and minimum)

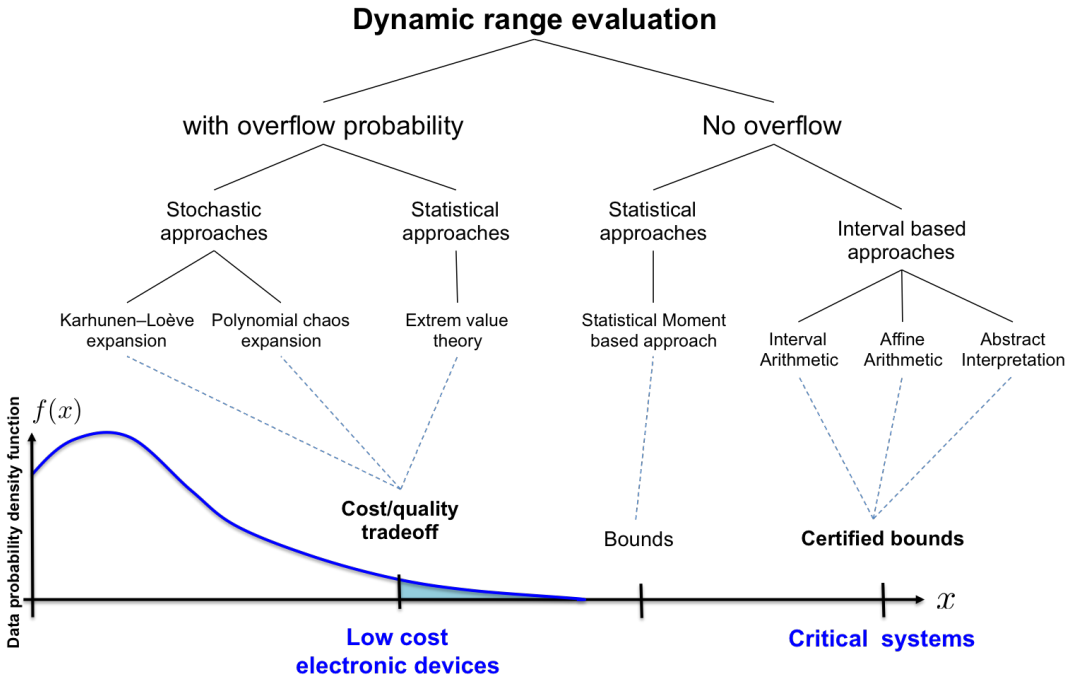


Figure 2.7 – Classification of the approaches for the dynamic range determination (from [2])

of a variable are known, the minimum integer word-length is computed in two's complement representation as follows

$$I_{WL} = \lceil \log_2(max|x|) \rceil + \alpha \tag{2.17}$$

with

$$\alpha = \begin{cases} 2 & \text{for } mod(\log_2(x_{MAX}), 1) = 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.18}$$

Figure 2.7 presents a classification of the existing techniques used to evaluate the dynamic range. When the system does not tolerate any computational error (critical system), the integer part word-length has to cover the entire range of possible values and any overflow occurrence may lead to a serious quality degradation. In this case, the data bounds should be determined by techniques that guarantee the absence of overflow occurrence. For example, techniques based on interval arithmetic satisfy this constraints, but at the expense of an overestimation of the data bounds. Statistical approaches that determine bounds from a set of simulation results can reduce the overestimation, but can not ensure the absence of overflow occurrence.

It should be noted that overflows occur when the number of bits of the integer part is not sufficient. Overflow occurrence degrades the system performance. However, the hardware implementation cost is unnecessary increased if the number of bits exceeds the needs.

Many applications are tolerant to overflows if the probability of overflow occurrence is low enough. In this case, determining the number of bits of the integer part is a trade-off between

the implementation cost and the system performance degradation. This is translated into an optimization problem where the integer word-length of each variable of the system is reduced while maintaining an overflow probability lower than the accepted probability. The challenge is to estimate the probability density function (PDF) of the data in order to be able to compute the overflow probability. Firstly, statistical approaches model the data PDF queue from a set of simulation results. Secondly, stochastic approaches model the variable PDF by propagating the data PDF model to the inputs inside the application.

2.2.1.1 Dynamic range evaluation without overflow

In the following part, the existing techniques used to determine the dynamic range while avoiding overflow occurrence are presented. These techniques are based on the estimation of the dynamic range for each variable using its extreme values.

Simulation based methods The statistical methods allow to estimate the dynamic range of a variable using its characteristics, determined by floating-point simulations. The simplest estimation of the dynamic range is to determine the maximum absolute value of the samples obtained during the simulation [36]. However, the quality of the estimation depends on the choice of the input stimuli and on the total number of samples.

Kim and Sung proposed to use a function of the mean and the standard deviation rather than the maximum absolute value [37]. This can be more pessimistic, but has the advantage of being less sensitive to the number of samples. The statistics collected for each variable x are the mean $\mu(x)$ and the standard deviation $\sigma(x)$. The authors then propose to choose a dynamic range R using the following expression

$$R(x) = |\mu(x)| + n(x).\sigma(x) \tag{2.19}$$

where n is an empirical weighting coeffecient [37, 38]. Estimating the dynamic range based on the average and the standard deviation rather than the maximum absolute value provides a better estimation of the dynamic range with fewer samples. However, this approach can be applied only if the distribution of x follows a uni-modal law. This is verified in [37] where the model is extended by measuring the coefficients of skewness and kurtosis of each variable which allows the classification of the signal as uni-modal or multi-modal, symmetrical or non-symmetrical with zero mean or non-zero mean. These classifications are then used to perform the scaling of more complex signals.

Interval arithmetic Interval arithmetic (*IA*) methods are based on associating each variable with an interval instead of a value [39]. The bounds of a variable at the output of an operator are calculated using the bounds of the input operands and the rule of the corresponding operation. Let $D_x = [\underline{x}, \overline{x}]$ and $D_y = [\underline{y}, \overline{y}]$ be the definition intervals of respectively the variables x and y. If

x and y are not correlated (x and y may take any value of their definition interval independent of each other), then:

$$D_{x+y} = [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \tag{2.20}$$
$$D_{ax} = [a\underline{x}, a\overline{x}] \quad \text{if} \quad a \geq 0 \tag{2.21}$$
$$= [a\overline{x}, a\underline{x}] \quad \text{if} \quad a < 0 \tag{2.22}$$
$$D_{x*y} = [min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}), max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})] \tag{2.23}$$
$$D_{-x} = [-\overline{x}, -\underline{y}] \tag{2.24}$$
$$D_{x-y} = [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \tag{2.25}$$
$$D_{x^2} = [min(0, |\underline{x}|, |\overline{x}|)^2, max(|\underline{x}|, |\overline{x}|)^2] \tag{2.26}$$

Interval arithmetic guarantees the absence of overflow in the algorithm if the input data values are in the definition domain used by the estimator. It computes the variable ranges at the compilation time and is not data dependent. However, the performance of this estimator is low because the method is conservative. The estimation is based on an analysis that considers the worst case scenario, which could result in overestimated bounds. Moreover, this method is often pessimistic because it does not take into account any correlations between the input operands, it considers that all the signals are independent and may take any value in their given interval. But not all the values in the interval are truly possible if there is a correlation between the operands. Thus, in this case, the method will provide overestimated bounds. In addition, it cannot be applied on recursive systems even in the case of a stable system.

Multi-Interval Arithmetic (*MIA*) is proposed as an improvement of the IA method [40, 41]. The enhanced method splits each interval into P disjoint subintervals

$$[x_{min}, x_{max}] = \bigcup_{i=1}^{P} [x_{i_1}, x_{i_2}] \tag{2.27}$$

The operations are performed on smaller intervals and the total dynamic range is determined by merging all the intermediate intervals. Thus, the dimensions of the final results are reduced in comparison to the traditional IA method. As IA, MIA guarantees the absence of overflow and underflow and does not address the correlation problem.

Affine arithmetic One of the proposed solutions to solve the dependency problem in IA is the affine arithmetic (AA) method [42, 43, 44, 45]. The authors extend the classical IA-based method by integrating the source and the sign amplitude of all uncertainties ϵ_i. The uncertainty of a variable x is represented by a linear expression given by:

$$\hat{x} = x_0 + x_1\epsilon_1 + \ldots + x_n\epsilon_n\epsilon_i \in [-1, 1], \tag{2.28}$$

ϵ_i is an independent source of uncertainty or error, which is added to the total uncertainty of the variable $\hat{x}$. Moreover, each source of uncertainty ϵ_i is weighted by a partial deviation x_i. The

definition interval of a variable x represented with an affine form is determined as:

$$x \in [x_{min}, x_{max}] = [x_0 - r_x, x_0 + r_x] \tag{2.29}$$

$$\tag{2.30}$$

with

$$r_x = |x_1| + |x_2| \ldots + |x_n| \tag{2.31}$$

The most important property of the AA method is that a noise coefficient can be shared between variables, which allows keeping track of the first order correlation (also called spatial dependency) between them. However, the temporal correlation between the value of a signal and its previous values is not taken into account. Thus, this technique becomes less effective in the field of Digital Signal Processing. Similar to IA, the dynamic range is propagated through affine operations. This step is straightforward for all linear operations as they preserve the affine property for the result as depicted in Equation 2.32.

$$\begin{aligned} \hat{x} &= x_0 + x_1\epsilon_1 + \ldots + x_n\epsilon_n \\ \hat{y} &= y_0 + y_1\epsilon_1 + \ldots + y_n\epsilon_n \\ \hat{x} + \hat{y} &= x_0 + y_0 + \sum_{i=1}^{n} (x_i + y_i)\,\epsilon_i \end{aligned} \tag{2.32}$$

For non-affine operations, the result is no longer a linear function of ϵ_i, and a linear approximation of the function is used in this case. The non-linear terms are bounded by a constant and added to the final affine expression. This results in loss of information and oversized bounds. For example, the multiplication is realized as in Equation 2.33. Other non-affine operations can be be considered [44].

$$\hat{z} = \hat{x}\hat{y} = \left(x_0 + \sum_{i=1}^{n} x_i\epsilon_i\right)\left(y_0 + \sum_{i=1}^{n} y_i\epsilon_i\right) \tag{2.33}$$

$$\hat{z} = (x_0 y_0) + \sum_{i=1}^{n} (x_0 y_i + y_0 x_i)\,\epsilon_i + z_k\epsilon_k \tag{2.34}$$

$$with \quad z_k = \sum_{i=1}^{n} |x_i| \cdot \sum_{i=1}^{n} |y_i| \tag{2.35}$$

The number of noise variables increases with each non-linear operator. The independence between these uncertainties would result in a loss of correlation between the signals and the range would explode for large computation data-paths.

L_p norm and transfer function based methods In [46, 47], the authors propose a methodology that calculates the dynamic range for Linear Time-Invariant (*LTI*) systems. The methodology uses

the L_1 norm and the concept of transfer function. Let's consider a linear system S with input x and output y, the output y at instant n depends on the input $x(n)$, on the N_e precedent input values $x(n-i)$ and on N_s precedent output values $y(n-i)$ (Equation 2.36).

$$y(n) = \sum_{i=0}^{N_e} b_i x(n-i) - \sum_{i=1}^{N_s} a_i y(n-i) \tag{2.36}$$

The objective of this approach is to transform the recurrent equation of the system (2.36) so that the output y is expressed in terms of the input x and its delayed versions. The first technique uses the impulse response h of the system. Thus, the output is the convolution between the input x and the impulse response h

$$y(n) = h(n) * x(n) \tag{2.37}$$

By using the L_1-norm [48], the maximum absolute value of y is

$$\max_n(|y(n)|) \leq \max_n(|x(n)|) \cdot \sum_{m=-\infty}^{+\infty} |h(m)| \tag{2.38}$$

If the maximal and minimal values of the input are known, the dynamic range can be computed for every variable in the system. This method can be used for any type of input signal and gives theoretical bounds for the output guaranteeing the absence of overflow. However, it only takes into consideration the maximum values and not the signal statistics. This L_1-norm gives conservative results.

The second technique uses the frequency response of the filter $H(e^{j\Omega})$. This approach, called standard Chebychev, determines the maximum absolute value for a given narrow-band input signal. Thus, the input signal is modeled by a sinusoidal function. The relationship between the maximum absolute value of y and x is as follows

$$\max_n(|y(n)|) \leq \max_n(|x(n)|) \cdot \max_\Omega(|H(e^{j\Omega})|) \tag{2.39}$$

2.2.1.2 Dynamic range evaluation in presence of overflow

When the signal has high variation in its amplitude throughout the execution, determining the word-length becomes a difficult task. Ensuring the theoretical range leads to a significant increase of cost. To reduce the cost and the execution time, the word-length of the integer part can be reduced so that the interval of variation is not entirely covered. As a consequence the overflow occurrence is authorized with a constraint regarding their probability of appearance. Variables that have long tailed PDFs can be approximated with tight intervals that correspond to a desired coverage probability.

This section presents the methods used to determine the dynamic range for a given overflow probability. The methods are based on deterministic procedures that provide theoretical results

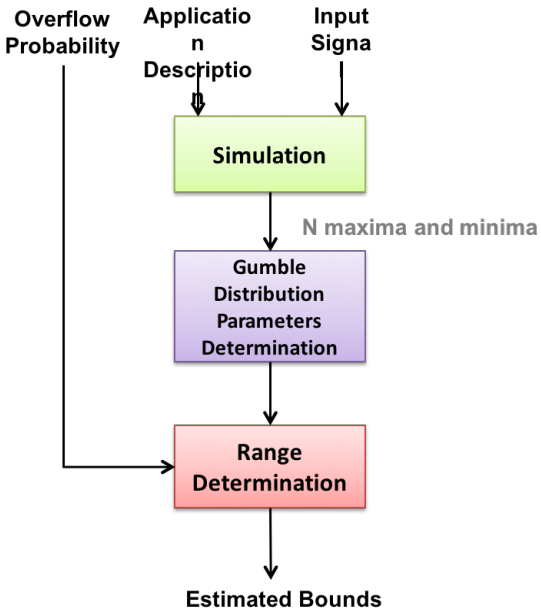


Figure 2.8 – Range determination methodology using Extreme Value Theory.

using a description of the input variability.

Extreme value theory The extreme value theory [49] is a statistical analysis based method concerned with the extreme deviations from the mean of the probability density function. Its purpose is to give a theoretical description of the distribution of the extreme values. This can be used to model the probability and magnitude of rare events. In [49], the author showed that the maxima and minima obtained for different realization of a random process follow the Gumbel distribution:

$$f(x) = \frac{1}{\beta} e^{\frac{-(x-\mu)}{\beta}} e^{-e^{\frac{-(x-\mu)}{\beta}}} \tag{2.40}$$

$$\beta = \frac{\sigma_x \sqrt{6}}{\pi} \tag{2.41}$$

$$\mu = \mu_x - \beta_\gamma \tag{2.42}$$

where μ_x and σ_x are respectively the mean and standard deviation of x. γ is the Euler's constant (≈ 0.5772). The Extreme Value Theory has been applied to the range estimation problem in [16, 15, 50] as shown in Figure 2.8. They use lightweight simulations for statistical data analysis that provides theoretical probabilities for an overflow event. The program is simulated N times, which leads to the extraction of N minima and maxima. Then, the parameters of the Gumbel distribution are estimated using the obtained results. Let P_r be theprobability that x is in the range $[x_{min}; x_{max}]$. Let P be the overflow probability, which is the complementary event of P_r. The maximum value x_{max} is derived from Equation 2.40 and results in [50]

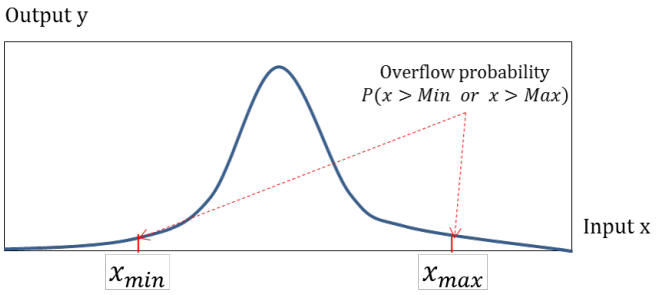


Figure 2.9 – Compute the range of a variable from its PDF.

$$x_{max} = \mu - \sigma_x \ln\left(\ln\left(\frac{1}{P_r}\right)\right) = \mu - \sigma_x \ln\left(\ln\left(\frac{1}{1-P}\right)\right). \tag{2.43}$$

Expression 2.43 determines the definition domain of the data for a given probability of overflow P defines by the developer. The larger the number of samples N is, the more accurate the statistical analysis becomes. The number of samples that should be provided for an application is determined empirically. If the sample size is not large enough, all the possible execution traces in the program are not covered. It should be noted that this method can be applied on any kind of system and outperforms affine arithmetic based methods in the range estimation and the area reduction especially for non-linear applications [50].

Stochastic methods Stochastic approaches aim at obtaining a complete representation of the variability of a variable x. The range of all the variables are represented by their PDF and obtained by propagating the variability characterization through the system. The range of all the variables is computed from the PDF with respect to a coverage probability as shown in Figure 2.9.

A general methodology to determine the dynamic range for a fixed overflow probability is presented in Figure 2.10 (proposed in [3]). This methodology is based on stochastic modeling of signals to determine the PDF of the output. The first step is to model the input signal by decomposing it in terms of several stochastic parameters. The second step calculates the stochastic parameters of the output, which allow to determine its PDF in a next step. It should be noted that several approaches can be used to determine these parameters. Finally, the dynamic range of the output is determined according to the authorized probability of overflow.

In [18, 20], a new approach for range estimation is presented. The method takes advantage of both the random and temporal dimensions that characterize the uncertainty of data in signal processing applications. The signal is modeled in the form of a stochastic process to determine its dynamic range. The input signals are modeled as a weighted sum of random variables. Then, the parameters associated with each input are propagated in the system to obtain the corresponding output parameters and probability density function. This allows the possibility of obtaining the dynamic range for a given probability of overflow.

In [18], the proposed method uses the Karhunen-Loeve Expansion (KLE) to decompose the

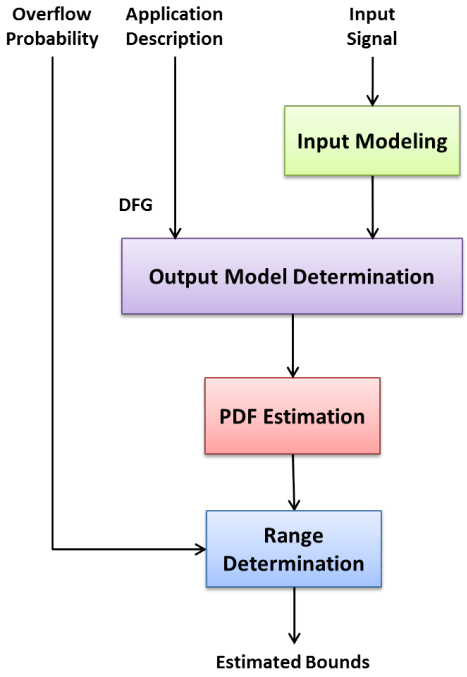


Figure 2.10 – Probabilistic range determination methodology [3]

LTI system input, modeled as an arbitrary random process, into k deterministic signals. Each signal is multiplied by a random variable. Starting from a description of the system behavior the KL expansions for the random processes corresponding to all variables can be generated. Based on this, full statistical information about the variables can be obtained by simulating the system (executing the program)k times. A stochastic process $p(t)$ is expressed as

$$p(t) = \sum_{i=0}^{\infty} \sqrt{\lambda_i} f_i(t) \mu_i \tag{2.44}$$

where $f_i(t)$ are the eigenfunctions, λ_i is the eigenvalues of the covariance function R, and μ_i are non-correlated random variables with unity variance and zero mean.

In contrast to the previous range representations, the KLE is a complete statistical description of the input process that can be used to determine the statistical moments or the entire probability distribution. Using the superposition property of the LTI systems, it is possible to determine the corresponding KLE description of the output using a limited number of simulations. However, the simulation period must be set long enough for the statistics of the responses to reach the steady state. The method fully considers both the spatial and temporal correlation. It can construct random process models from real data or empirical statistics, instead of using oversimplified models or assumptions.

For non-linear systems, superposition cannot be applied anymore. Thus, the authors of [20] proposed the use of the Polynomial Chaos Expansion (*PCE*). The authors show how the PCE of the input is obtained from the corresponding KLE representation using a projection method. Using

the PCE arithmetic, the variability of the input can be statically propagated in non-linear systems and the PCE representation for all the variables can be obtained in any type of system. The PCE of a 2^{nd} order stochastic processe $X(\Theta)$ is expressed as follow

$$X(\Theta) = \sum_{i=0}^{\infty} \alpha_i \Psi_i(\zeta(\Theta)) \tag{2.45}$$

where α_i are constant coefficients, $\zeta = (\zeta_1, \zeta_2 ... \zeta_d)$ is a set of d independent second order random variables and $\Psi_i(\zeta)$ is multidimensional orthogonal polynomials.

Furthermore, the authors propose a word-length optimization criterion under SNR constraints[20]. However, when the overflows occur during the computation, this evaluation may become inaccurate.

In [3], a probabilistic approach for the dynamic range evaluation has been developed using KLE to represent the variability of the input signal in linear-time invariant systems and PCE in non linear systems. Compared to the previous method, the variability of the input is statically propagated through the data-flow graph and the analytical representation of the output is obtained. The range is further computed from the PDF with respect to a coverage probability.

2.2.2 Fractional word-length optimization

Computation in fixed-point arithmetic has limited accuracy and generates quantization error at the output. The quantization error is considered as a noise added to the result and evaluated by the difference between the output in finite and infinite precision [29]. It is therefore necessary to verify that the behavior of the algorithm using fixed-point arithmetic is modified within a reasonable limit. Thus, the fractional part is determined based on a trade-off between the needed accuracy and the circuit cost. There are various metrics to evaluate the numerical accuracy. The most commonly used metric in digital signal processing domain is the power of the quantization noise or the signal to quantization noise ratio (*SQNR*). Let P_b be the quantization noise power and P_y the output signal power, the SQNR in dB is defined by

$$SQNR_{dB} = 10 \log \frac{P_y}{P_b} \tag{2.46}$$

An alternative metric can be the error bounds $e \in [e_{min}, e_{max}]$ [51]. It is used to ensure an absolute maximal value of the quantization noise. A third metric corresponding to the number of significant bits associated with a variable [52] can be used. This method estimates the number of bits of a given variable that are not modified by the surrounding noise and thus represent correctly the data. For an example of a $16-$bit data, the number of significant bits not modified and allowing a correct representation of the signal is 13 when the quantization noise change the values of the last three bits.

To determine the quantization noise power at the output of the algorithm two approaches may be used. The first approach is to determine the statistical parameters of the quantization error

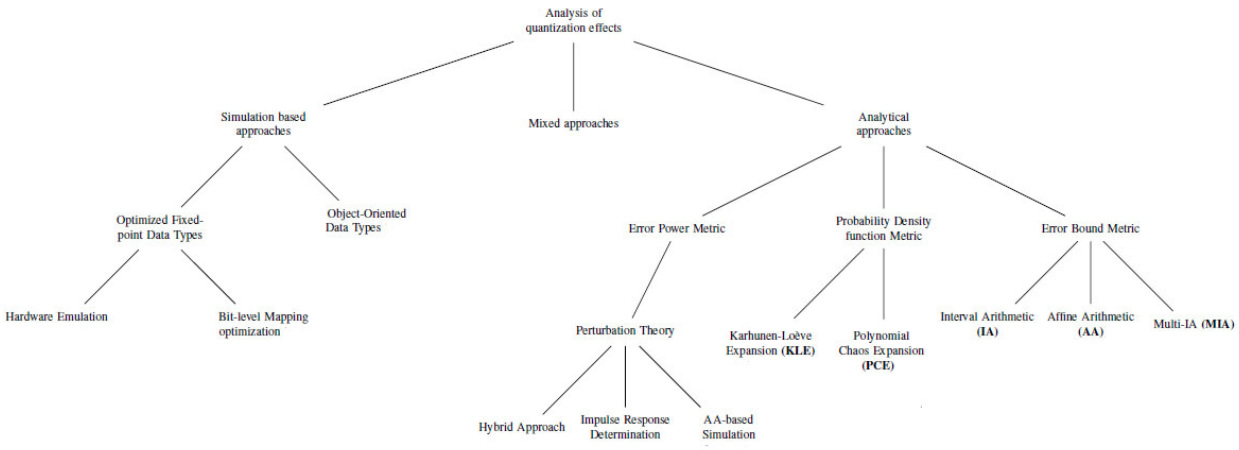


Figure 2.11 – Classification of the approaches used to analyze the effects of the quantization noise

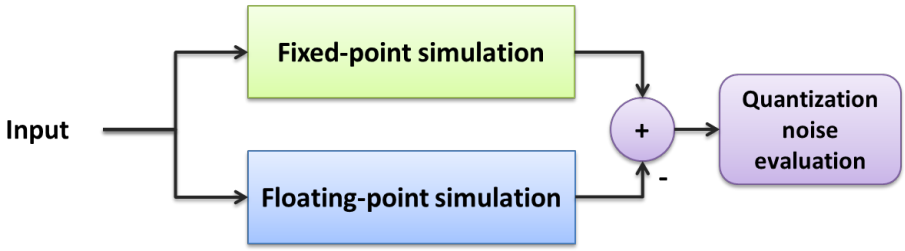


Figure 2.12 – Simulation-based computation of quantization error

from the fixed-point and floating-point simulations of the algorithm. The second determines the analytical expression of the quantization noise power by propagating a noise model in the flow chart of the algorithm. Figure 2.11 presents a classification of the methods used to analyze the quantization noise.

2.2.2.1 Simulation based methods

The evaluation of the performance of a fixed-point system using simulation methods is based on evaluating the output of the system using bit-true fixed-point simulation to obtain the value of the quality criterion associated with the application. The advantage of these methods is their ability to provide accurate results for any type of system. The accuracy of the fixed-point specification is evaluated statistically from the signals obtained by simulating the algorithm in fixed-point and floating-point. The floating-point simulation output is close to the output of infinite precision simulation because of its low quantization error compared to the one obtained with fixed-point simulation. Thus, floating-point simulation is considered as the reference and the power of the quantization noise is directly obtained from the second order moment of the difference between the two simulations. The approach is presented in Figure 2.12 and the power of the quantization

noise is evaluated as in Equation 2.47, where y and y_{fixed} are respectively the output results of the floating-point and fixed-point simulations.

$$P_b = E[(y - y_{fixed})^2] \tag{2.47}$$

To get accurate results, a high number of samples, denoted N_{ech}, at the input of the algorithm has to be used. Let N_{ops} be the number of operations defined in the description of the algorithm and N_i be the number of time that the accuracy is evaluated during the optimization process. A first estimation of the number of points (N_{pts}) to be calculated, presented in Equation 2.48, shows the need for an effective simulator to obtain a reasonable simulation time.

$$N_{pts} = N_{ech} \cdot N_{ops} \cdot N_i \tag{2.48}$$

Various methods have been proposed to emulate the mechanisms of fixed-point arithmetic. The method presented in [53] can simulate a fixed-point specification, using the concepts of overloading operators in $C++$ language level. The operator overloading makes the implementation of the fixed-point algorithm significantly longer than a floating-point simulation. The same concepts are implemented in fixed-point simulation using SystemC [54]. In [55], the floating-point and fixed-point simulation time are compared for some signal processing applications. Let R_{sim} be the ratio between the simulation time $t_{sim-fxpt}$ obtained with fixed-point data types and the simulation time $t_{sim-flpt}$ obtained with floating-point data types

$$R_{sim} = \frac{t_{sim-fxpt}}{t_{sim-flpt}} \tag{2.49}$$

The average ratio R_{sim} is equal to 540 for standard SystemC fixed-point data types and 120 for limited precision SystemC fixed-point data types. The data type `pfix` is proposed in [56] to reduce the simulation time of the method described above. It aims at improving the computation time of fixed-point operations by using efficiently the floating-point data types of the host machine. The fixed-point simulation time of a fourth order IIR filter with `pfix` type leads to a ratio R_{sim} of 7.5.

The fixed-point simulation can be accelerated by executing it on a more adequate machine like a fixed-point DSP [57, 58, 55], or an FPGA [59] through hardware acceleration. The method presented in [58, 55] uses integer types in the machine for a more efficient encoding of the data in order to reduce the execution time of the simulations. This concept is also used in the simulator HYBRIS associated with Fridge tool [60]. These methods reduce the execution time of the fixed-point simulation. However, the effort required to optimize fixed-point data has not been quantified. Indeed, the optimization techniques described above are relatively complicated, and a new simulation is required every time the fixed-point specifications change.

2.2.2.2 Analytical approach

The objective of the analytical procedures is to define a mathematical expression of the accuracy metric according to the data word-length. Obtaining this accuracy metric expression can take time. However, once this expression has been determined, the accuracy can be evaluated from the mathematical function and this evaluation process is fast.

Error power evaluation The existing approaches to calculate the analytic expression of the quantization noise power are based on the perturbation theory. The finite precision signal is modeled as the sum of the infinite precision signal and a perturbation of a very small amplitude compared to the infinite precision signal amplitude. This perturbation is assimilated to an uniformly distributed white noise, which is uncorrelated with the signal or any other quantization noise in the system. Each noise source b_i propagates within the system and contributes to the system output noise. This propagation has to be modeled to obtain an expression of the output quantization noise. A first-order Taylor series expansion [61, 62] is used to linearize the behavior of the operations in the presence of an input quantization noise. In [63, 64], the propagation of the quantization noise is modeled using affine arithmetic. These models based on the perturbation theory are only valid for smooth operations.

The output noise b_y is the sum of the contributions of all the N_e noise sources. The second order moment of b_y can be expressed as the weighted sum of the statistical parameters of the noise sources [65]

$$E(b_y^2) = \sum_{i=1}^{N_e} K_i \sigma_{b_i}^2 + \sum_{i=1}^{N_e} \sum_{j=1}^{N_e} L_{ij} \mu_{b_i} \mu_{b_j} \tag{2.50}$$

The terms μ_{b_i} and $\sigma_{b_i}^2$ are respectively the mean and the variance of the noise source b_i. The terms K_i and L_{ij} are constant terms. These terms depend on the system located between the noise source b_i and the output. Thus, these terms are evaluated only once to obtain the analytical expression of the precision.

In [65], Linear Time Invariant systems are considered. The coefficients K_i and L_{ij} are computed from the impulse response of the LTI system between the noise sources and the output. This method has been extended in [66] to also handle non-linear systems (including recursive systems).

Another method based on affine arithmetic simulation is proposed for LTI systems in [64]. An affine form is assigned to each noise source. Then, the coefficients K_i and L_{ij} are determined from the affine form of the output noise. The same method has been proposed for non-LTI systems in [63]. However, the method might lead to a too huge number of terms when the number of iterations is large.

In [67, 68], hybrid techniques combining analytical expressions and simulations are proposed. The coefficients K_i and L_{ij} are computed by solving a linear system of equations having K_i and L_{ij} as variables.

Error bounds evaluation Several techniques have been suggested to evaluate the bounds of the quantization noise. Bit-true simulation is the most extended techniques used to estimate the quantization noise bounds [69, 70]. It is usually combined with other techniques to reduce the simulation time. For example, statistical refinement and bit-true simulations are combined in [50, 71], where the authors apply a technique based on the Extreme Value Theory with Adaptive Simulated Annealing to determine the quantization noise bounds of the output.

Interval-based methods can be used to evaluate the the errors bounds. Indeed, each error introduced by quantization process with rounding or truncation modes is bounded. Since the noise sources are modeled as independent input signals, the techniques used for range analysis can also be used for the computation of the noise bounds. Interval arithmetic and affine arithmetic techniques [72] already presented in Section 2.2.1.1 can be used to determine the bounds of the output error of the system. An interval is associated to each noise source generated during the quantization process. Then, these intervals are propagated through the system. In [73], the authors used a Multi-interval arithmetic (*Multi-IA*) approach to provide refined bounds and reduce the overestimation of IA.

Interval-based computations can be applied to analyse the precision in both fixed-point [43] and floating-point [51] systems. However, direct use of these techniques is limited to non-recursive systems. As mentioned before, the problem with AA is the support of non-affine operations and its weak treatment of correlation.

A new method is proposed by Wadekar and Parker [74], where perturbation theory is exploited to evaluate the noise bounds. The authors proposed a propagation model based on Taylor series decomposition of functions with two inputs. Automatic differentiation is proposed by Gaffar et al. in [75, 76] for linear or quasi-linear systems. An approach that uses Arithmetic Transformations (*AT*) has been proposed in [77, 78]. The AT representation guarantees the accurate propagation of the polynomial terms due to its canonical form. This approach has been extended to evaluate systems with feedback loops in [79, 80].

Propability density function One of the essential metrics used to analyze the effect of signal quantization is the probability density function (PDF) of the quantization noise. In comparison to the quantization error bounds or quantization noise power, which are suitable for differential operations, PDF provides more information and allow better analysis of unsmooth operations. In [19], the quantization noise PDF is determined using the Karhunen-Loeve Expansion representation of the quantization noise. Using the superposition property and the transfer function of the system, the KLE description is computed by propagating the KLE coefficients associated with noise sources. In [81], the authors propose a stochastic approach based on a combination of Modified Affine Arithmetic (*MAA*) and Polynomial Chaos Expansion to determine the output quantization noise PDF. The advantage of the PCE representation is its applicability to non-linear operations. The output quantization noise of a smooth system is modeled as a weighted sum of a Gaussian random variable and a uniform random variable as in [82] and as a generalized Gaussian random variable in [83]. The effect of quantization noise on an unsmooth operator like signum function and Quadrature Amplitude Modulation constellations diagrams is studied in [84] and [5].

When the system include several unsmooth operations, the evaluation of the quantization effect using purely analytical approaches becomes an issue. In [6], a hybrid approach is proposed. This approach uses analytical models to accelerate quality evaluation based on simulation. The idea in the hybrid approach is to simulate parts of the system only when un-smooth errors occur. Otherwise, analytical results based on perturbation theory are used. This concept will be extended in our approach and detailed in Chapter 6.

2.3 ID.Fix

The ID.Fix EDA tool has been developed at IRISA/INRIA labs since 2008 [4], and the goal of this thesis is to extend the capabilities of this tool by providing techniques to evaluate the application quality when overflows or unsmooth errors occur. The ID.Fix EDA tool allows the conversion of a floating-point C source code into a C code using fixed-point data types. The tool is implemented using the GECOS framework [85], a compiler infrastructure that performs source to source transformations and works with C or C++ code. The signal flow graph of the algorithm is extracted by inference [86]. The analytical expression of the loss of accuracy obtained at the output is evaluated by a fully automatic analytical technique developed in [65, 62]. The cost metric is the sum of individual operator cost. The modular nature of the tool and the GECOS platform allows to perform experiments with optimization heuristics. The synoptic of ID.Fix infrastructure is presented in Figure 2.13.

2.3.1 ID.Fix tool description

2.3.1.1 Tool functionality

The purpose of ID.Fix is the fixed-point conversion of a floating-point C source code that describes the signal processing application. The conversion process defines the optimized fixed-point format of the operation operands. The integer part word-length is determined to avoid the overflow occurrence or limit its probability, and the fractional part word-length is determined to minimize the implementation cost respecting the accuracy constraint $P_{b_{max}}$. The infrastructure of ID.Fix is made-up of three main modules corresponding to fixed-point conversion (Fix.Conv), accuracy evaluation (Acc.Eval) and dynamic range evaluation (Dyn.Eval).

2.3.1.2 Tool inputs

The inputs of the tool for the fixed-point conversion of an application are:

— App.c: The input source code of the application written in C language with floating-point data types. Only a subset of the C language is supported to describe the application. Currently, pointers and structures are not supported.

— $P_{b_{max}}$: Quantization noise power of the application output is expressed in dB. It corresponds to the accuracy constraint for the optimization process of the operators word-length.

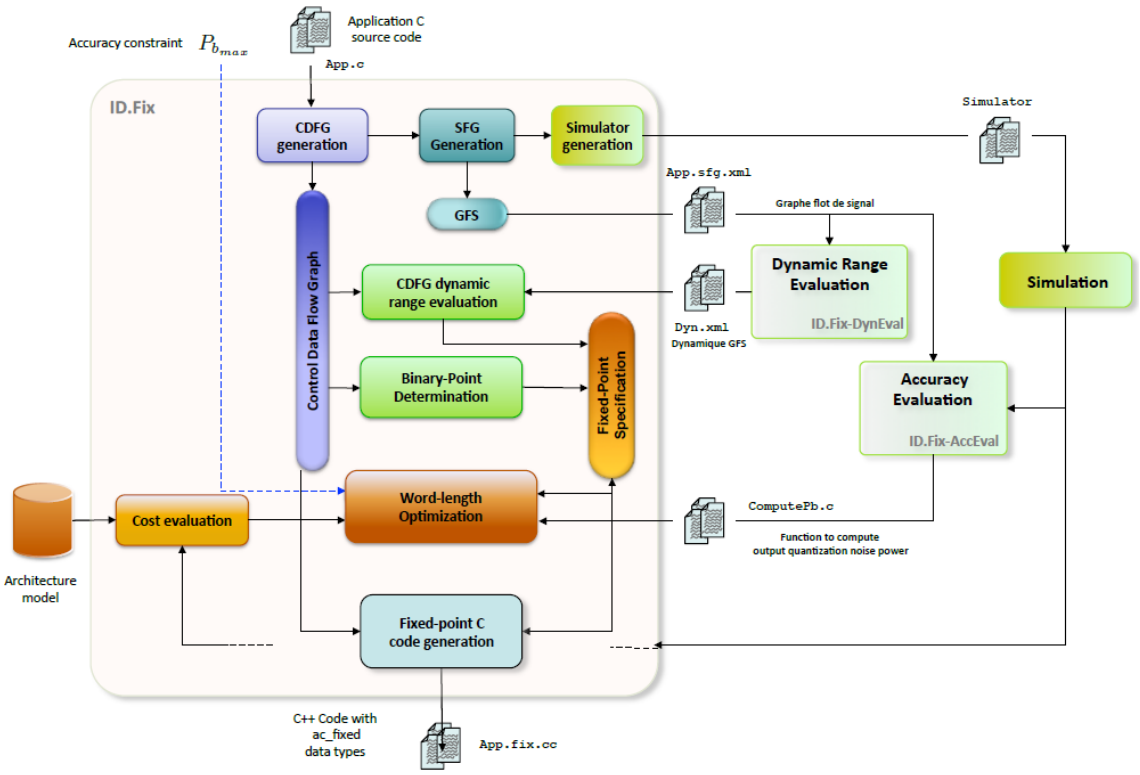


Figure 2.13 – Detailed flow of the EDA tool ID.Fix [4]

The evaluation of this accuracy metric is detailed in Section 2.2.2.

— App.Arch.xml: XML file that defines the architecture model to estimate the implementation cost during the optimization process. The model includes all the words-length supported by the architecture and the associated implementation cost.

Pragma directives are also taken as input in order to specify some parameters for fixed-point conversion (dynamic range, inputoutput word-length, delay operations).

2.3.1.3 The tool output

The output of the tool is the `App.fix.cc` file containing the source code of the application in fixed-point types such as Mentor Graphics `ac_fixed` or SystemC `sc_fixed`. The fixed-point specification is optimized according to the types supported by the architecture and the accuracy constraints.

A given variable var of `ac_fixed` type is declared with the following expression

$$\texttt{ac\_fixed<}\ w_D, w_{IP}, S, Q, O\ \texttt{>}\ \texttt{var},$$

where w_D is the total number of bits, w_{IP} is the number of bits of the integer part, S is a boolean type indicating whether the data is signed, Q is the quantification mode and O the overflow mode (saturation or wrap-around). The advantage of ac_ fixed type is the ability to specify all the fixed-

point parameters of a given variable with the format change operations (cast) and data alignment. However, the fixed-point parameter are static and can not change over time.

2.3.2 Tool description

The fixed-point conversion tool ID.Fix-Conv is developed in the compilation infrastructure Gecos (Generic Compiler Suite). The front-end of Gecos generates the intermediate representation on which various transformations are performed. The tool developed within Gecos consists of two branches. The first one contains necessary transformations for the fixed-point conversion and the regeneration of the C source code with fixed-point types. From the C source code, an intermediate representation is generated with the GECOS front-end. This intermediate representation, used for the fixed-point conversion process, is a Control and Data Flow Graph (CDFG). The CDFG is an oriented graph where the nodes represent control blocks. Each control structure of the C language has a specific type of block. The data processing is included in the basic block (control nodes) and is made-up of a set of operation o_i and data d_i.

The second part generates the Signal Flow Graph (*SFG*) of the application. The SFG is used in the dynamic and accuracy evaluation modules. The dynamic and accuracy evaluation are performed on the SFG of the application. The SFG is generated by flattening the CDFG. Consequently each operation o_i is translated into several operations o'_j in the SFG. The indexes of all operations o'_j of SFG corresponding to the duplication of the operations o_i are grouped in the set T_i.

The operations o_i and the data d_i contributing to the output evaluation are indexed with a unique index i. This index i is then used to access to the fixed-point specifications associated with o_i and d_i. Three methods are available for the evaluation of the data dynamic range: interval arithmetic, L_1 norm and stochastic approach based on Karhunen-Loeve Expansion (KLE). Interval arithmetic based methods cannot be applied on recursive systems and the flow graph shall not contain cycle. Methods based on L_1 norm use the transfer function concept and are only valid on LTI systems. These two methods evaluate the dynamic range ensuring no overflow occurrence. The third method, which is based on KLE, allows the evaluation of the dynamic range of the LTI system output according to an overflow probability defined with a pragma. The KLE and L_1 norm based methods use the system impulse response. Thus, the dynamic range is evaluated through the SFG of the application and the same software infrastructure is used for the accuracy evaluation. The ID.Fix-DynEval tool determines the dynamic range of all the data d'_j of the SFG. The dynamic ranges of the data d_j of the CDFG are obtained from the dynamic ranges of d'_j.

After determining the binary point position, the data word-length are optimized. The data word-length optimization problem is expressed as the minimization of the implementation cost under accuracy constraint. The optimization algorithms implemented in the tool correspond to a greedy algorithm ($min + 1bit$), and a branch and bound search. The accuracy is evaluated using the C source code generated by the ID.Fix-accEval module, which allows the quantization noise power computation according to the word-length and the quantification mode used. The code is compiled, and the optimization algorithm is called through a Java Native Interface (*JNI*). The overall cost of the implementation is obtained from the cost associated to each operation. It is

calculated using the architecture database and the number of times the operation is performed. This information is determined by simulating (profiling) a set of input vectors.

The last step generate the `App.fix.cc` file, which contains the source code of the application with the fixed-point types determined in the previous steps.

2.4 Conclusion

After the description of floating-point and fixed-point representations, this chapter described the fixed-point to floating-point conversion process used to implement an application into a fixed-point architecture. The conversion process is an optimization problem divided into two parts corresponding to the determination of the integer part word-length and the fractional part word-length under application quality constraints. The fractional part word-length determines the numerical accuracy of the application, while the integer word-length determines the dynamic range and controls the occurrence of overflows. This chapter presented a review of the existing methods for both the range estimation and the accuracy analysis.

Based on the surveyed methods, analytical approaches, such as affine and interval arithmetic, allow fast evaluation of the application quality, but may overestimate the required word-length. Statistical approaches take into account the possibility of tolerating errors (overflow or un-smooth error). However, it is not always trivial to find the link between the probability of error occurrence and the application quality. Simulation based methods can be applied on any type of systems, but they are time consuming. Thus, reducing their simulation time is a key point in enhancing their applicability. To overcome the limitation of simulation based methods, a new framework is proposed in the next chapter. The proposed framework uses selective simulation to accelerate the evaluation of the application quality.

Most of the work is focused on optimizing the FWL while satisfying the accuracy constraint. Nevertheless, optimizing the IWL can significantly decrease the implementation cost when a slight degradation of the application performance is acceptable. Indeed, many applications are tolerant to overflows if the probability of overflow occurrence is low enough. Thus, the challenge is to provide an integer word-length optimization technique based on an efficient application quality evaluation. We propose a new approach and an associated framework using selective simulations to accelerate the simulation of overflow effects in chapter 3 and 4. Compared to complete fixed-point simulation based approaches, where all the input samples are processed, the proposed approach simulates the application only when overflow occurs. Indeed, overflows must be rare events to maintain the system functionality. Consequently, selective simulation allows reducing significantly the time required to evaluate the application quality criterion. Then, a new IWL optimization algorithm is proposed in Chapter 5. This algorithm efficiently exploits selective simulation based technique go accelerate the optimization of the IWL.

In a second time, we apply the framework of selective simulation to evaluate the accuracy in the case of fractional word-length optimization for system with un-smooth operations. Fractional word-length optimization is investigated in chapter 6.

CHAPTER 3

Selective Simulation Framework

3.1 Introduction

As shown in the previous chapter, reducing the word-length of fixed-point operations can significantly decrease the cost of hardware implementation. The reduction of the integer part word-length (IWL) can lead to overflows if the data dynamic range is not totally covered. Moreover, the reduction of the fractional part word-length (FWL) decreases the numerical accuracy and leads to a quantization noise. When the quantization step is too high, errors are considered un-smooth and have a high amplitude compared to the signal. Un-smooth errors and overflows have a huge impact on the quality of the application output. Thus, their probability of occurrence must be low enough to have an acceptable quality degradation.

In the fixed-point refinement process, the data word-lengths are optimized for a given quality constraint. Thus, the degradation of the application quality due to overflows or un-smooth errors should be evaluated at each iteration of the word-length optimization process. Several types of methods exist to evaluate the quality. Simulation-based methods have the advantage to support any type of systems. Nevertheless, when the number of iterations is high, the simulation becomes time consuming.

The probability of overflow and un-smooth error must be low. Thus, only few input samples lead to an error and contribute to the quality degradation. Existing methods for quality evaluation simulate the system for all the input samples. In this chapter, we propose a method to overcome the long simulation time in the case of small probability of overflow or un-smooth error. This method simulates the system to evaluate the quality only when a rare event (overflow or un-smooth error) occurs.

In the first section, the problematic of rare events is presented. Then, the framework of the proposed method is detailed. Finally, the approach is extended for the quality evaluation at the system level.

3.2 Context and motivations

To ensure acceptable application quality, overflows and un-smooth errors due to fixed-point representation should rarely occur. In this section, we explain the concept of overflow and its relation with the number of bits allocated for the integer part of a fixed-point data. Moreover, we explain the concept of un-smooth operation and its relation with quantization noise.

3.2.1 Overflow

An overflow occurs when the result of an operation requires a number of bits for the integer part of a fixed-point data greater than the allocated number of bits. In other words, the resulting value is out of the range of valid values, *i.e.* greater than the maximum representable value or lower than minimum representable value.

The reduction of the number of bits for the integer part of each variable increases the probability of overflow. To illustrate the relation between the overflow probability and the number of bits for the integer part, let's consider a random variable x of normal distribution with a probability density function (*PDF*) depicted in Figure 3.1 and expressed as

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp^{-\frac{x^2}{2}} \tag{3.1}$$

The integer word-length of x can be computed with the equation 2.17 given in page 16. For simplification, a symmetric definition domain for a variable x is considered and obtained by excluding the minimum value of the initial definition domain. In this case, the integer word-length of x is computed with

$$w_x = \lfloor \log_2 |x(n)| \rfloor + 1 + S \tag{3.2}$$

where S is equal to 1 for signed data and 0 for unsigned data. In the following we consider $S = 1$.

Let $P_{IWL}(w_x = w)$, be the probability that w bits are required for the integer part of the value x. This probability is computed with the following expression

$$\begin{aligned} P_{IWL}(w_x = w) &= P(\lfloor \log_2(|x|) \rfloor + 2 = w) && (3.3)\\ &= P(w - 2 \leq \log_2(|x|) < w - 1) && (3.4)\\ &= P(2^{w-2} \leq |x| < 2^{w-1}) && (3.5) \end{aligned}$$

The probability density function $f_w(w)$ of the integer word-length of x is given in equation 3.6 and presented in figure 3.2.

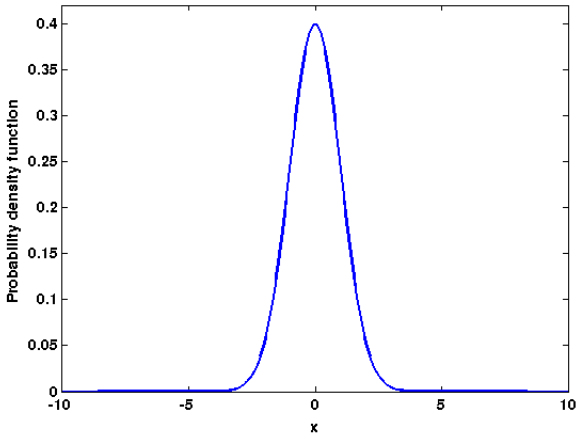


Figure 3.1 – Probability density function of variable x

$$f_w(w) = \int_{-2^{w-1}}^{-2^{w-2}} f(x)dx + \int_{2^{w-2}}^{2^{w-1}} f(x)dx \tag{3.6}$$

$$= \frac{1}{\sqrt{2\pi}} \left(\int_{-2^{w-1}}^{-2^{w-2}} \exp^{-\frac{x^2}{2}} dx + \int_{2^{w-2}}^{2^{w-1}} \exp^{-\frac{x^2}{2}} dx \right) \tag{3.7}$$

$$= \text{erf}\left(\frac{2^{w-1}}{\sqrt{2}}\right) - \text{erf}\left(\frac{2^{w-2}}{\sqrt{2}}\right) \tag{3.8}$$

Let w_0 be the number of bits allocated for the integer part of the variable x during computation. Each sample of x having an IWL higher than w_0 leads to an overflow. The probability of overflow occurrence is computed using $f_w(w)$ or $f(x)$ as follows:

$$P_{ovf}(w_0) = P(w_x > w_0) \tag{3.9}$$

$$= \sum_{w=w_0+1}^{+\infty} f_w(w) \tag{3.10}$$

$$= \int_{-\infty}^{-2^{w_0-1}} f(x)dx + \int_{2^{w_0-1}}^{+\infty} f(x)dx \tag{3.11}$$

If w_0 decreases, the number of samples leading to an overflow increases and quickly becomes very large as we can see in Figure 3.3. Table 3.1 provides the overflow probability for several integer word-length w_0 for the variable x.

The maximum overflow probability that can be tolerated depends on the criticality of the application. The more the application is critical the more the maximum overflow probability must be low or even null.

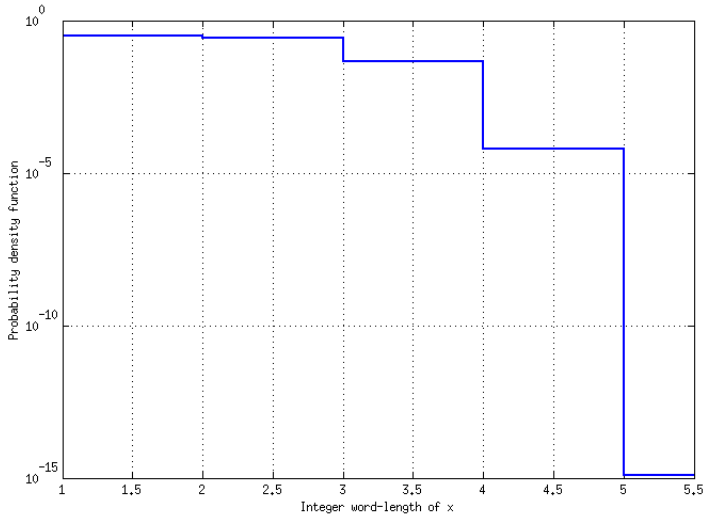


Figure 3.2 – Probability density function of integer word-length of variable x

$w_0(bits)$	$P_{ovf}(w_0)$
1	0.316
2	0.045
3	$6.3x10^{-5}$
4	$1.2x10^{-15}$

Table 3.1 – Overflow probability for different word-length w_0

3.2.2 Un-smooth Error

The second case of rare event considered in this work is un-smooth errors. After a presentation of the model for quantization process (reduction of the number of bits for the fractional part), the concept of un-smooth error is detailed.

Noise model In general, the effect of quantization process can be modeled as a noise. Using the Pseudo Quantization Noise model (*PQN*) [29], the effect of uniform quantization can be modeled with an additive white noise b_x. The quantized signal $\hat{x}$, is statistically equivalent to the sum of the original signal x and the noise b_x as shown in Figure 3.4. The noise power is determined from the quantization step-size q and the quantization mode. The noise b_x is uniformly distributed and is statistically uncorrelated with the quantized signal.

The quantization process is called smooth when the assumptions associated with the PQN model are satisfied. Especially, The quantization step-sizes must be small in comparison to the dynamic range of the signal. When the quantization step-size is large enough, PQN model is no longer valid and the quantization process is un-smooth. An operation is called smooth when the

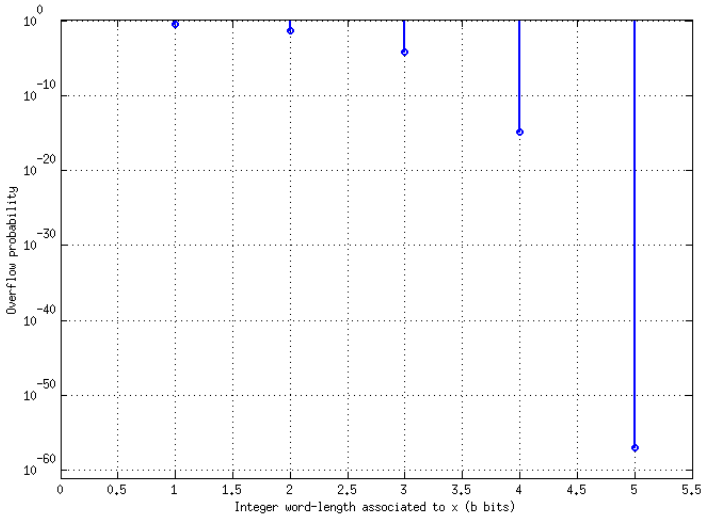


Figure 3.3 – Overflow occurrence with respect to IWL allocated for x

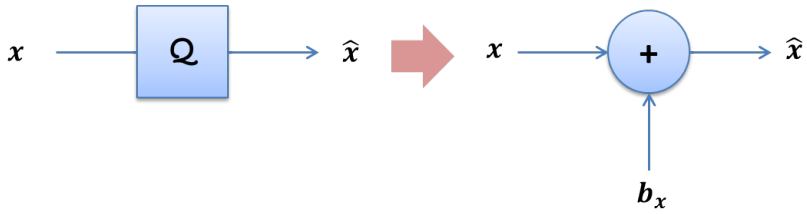


Figure 3.4 – Quantization noise model

change in the quantized input signal value causes a small change in the operation output.

Error Decision error is a good example of an un-smooth operation. Figure 3.5 shows the behaviour of a decision operation defined in equation 3.12 and the effect of the quantization of the input on the decision.

$$\begin{aligned} y = S_i \quad & \text{if} \quad x \in R_i \\ R_1 = \{x\} \quad & \text{such as} \quad x \leq a \\ R_2 = \{x\} \quad & \text{such as} \quad x > a \end{aligned} \tag{3.12}$$

The output of the decision operation is S_1 if the value of the input signal x is less than or equals to the threshold a and S_2 if the value of x is higher than a. The fixed-point conversion of x leads to $\hat{x}$ and adds a quantization noise e that perturbs the signal x. Let q be the maximum distortion $e \in [-q; q]$. When x has a value x_2 sufficiently far from the boundary ($|(|x_2 - a) > q$), the perturbation q is not large enough to make the signal cross the decision boundary. In this case, the outputs of both finite and infinite precision are S_2 and no decision error occurs. However, the perturbed signal may cross the boundary when x has a value x_1 such that $|(|x_1 - a) < q$. This

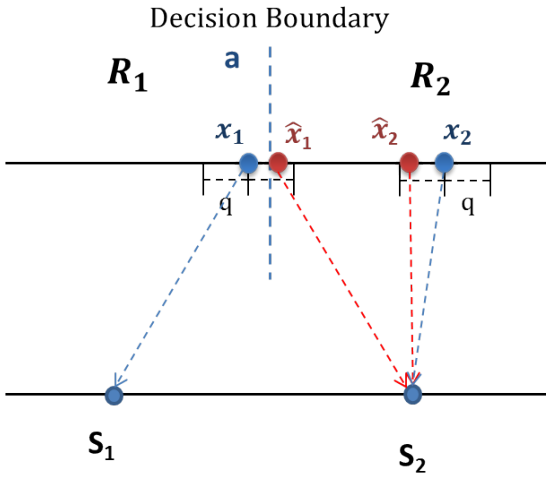


Figure 3.5 – Behaviour of a decision error operator in presence of perturbation.

may lead to an output S_2 instead of S_1, which represents a decision error.

un-smooth error probability To study the un-smooth error probability, we consider a system with one un-smooth operation U as shown in Figure 3.6. The decision error probability, due to an additive quantization noise, can be computed by comparing the fixed-point output $\hat{y}$ with the output obtained in infinite precision y.

Let's consider an instance of the input signal. The fixed-point signal $\hat{x}(n)$ is obtained by adding the quantization noise $b_x(n)$ to the input $x(n)$ as follow

$$\hat{x}(n) = x(n) + b_x(n) \tag{3.13}$$

When the value of the signal x is close to the decision boundary and the quantization noise is large enough, the perturbation can drift the input signal to cross the boundary leading to a wrong output (error). An error occurs when $x(n)$ and $\hat{x}(n)$ respectively leads to different values S_i and S_j with respect to the decision boundary. If $x(n)$ belongs to the region R_i, the probability $P_{i,j}$ of having a decision error, is defined with the following expression

$$P_{i,j} = P\left(\hat{x} \in R_j | x \in R_i\right) \tag{3.14}$$

The total error probability $P_{i,j}$ can be expressed as

$$P_{i,j} = P(x \in R_i \cap \hat{x} \in R_j) + P(x \in R_j \cap \hat{x} \in R_i) \tag{3.15}$$

In [21], the author evaluates P_e, the total decision error probability at the output of a decision operation having L regions with the following expression

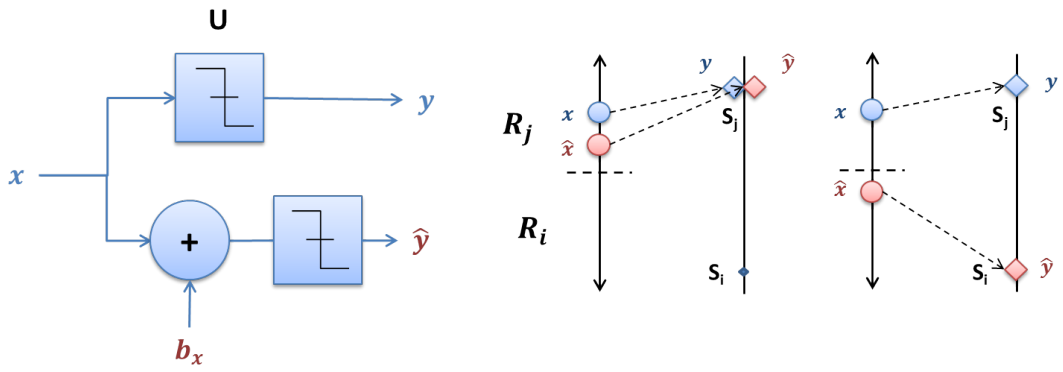


Figure 3.6 – Decision error due to input quantization

$$P_e = 1 - \sum_{i=1}^{L} P_{i,i} \tag{3.16}$$

and with

$$P_{i,j} = \int_{R_i} \int_{R_j} f_x(x) f_b(b-x) db.dx \tag{3.17}$$

$P_{i,j}$ is the probability of the decisions $y = S_i$ ($x \in R_i$) and $\hat{y} = S_j$ ($\hat{x} \in R_j$). The term $f_b(b)$ and $f_x(x)$ are the probability density functions of respectively the noise b_x and the infinite precision signal x.

Let's consider a decision operation with two decision outputs S_1 and S_2 and two input regions $R_1 =]-\infty, 0]$ and $R_2 =]0, +\infty[$. The input signal follows a normal distribution with a zero mean and a standard deviation equal to σ_s. The PDF of the input signal is equal to $f_x(x) = \frac{1}{\sqrt{2\pi}} exp^{-\frac{x^2}{2\sigma_s^2}}$. From [82], a normal distribution for the quantization noise f_b can be considered when this noise results from the contribution of several quantization processes. Its PDF is $f_b(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{x^2}{2\sigma^2}}$. Hence, the total decision error probability is evaluated as in equation 3.18, where P is computed as in equation 3.19.

$$P_e = P_{1,2} + P_{2,1} = 2.P \tag{3.18}$$

The probability P is equal to

$$P = \int_{-\infty}^{0} \int_{0}^{+\infty} f_x(x) f_b(b-x) db.dx \tag{3.19}$$

$$P = \int_{-\infty}^{0} \int_{0}^{+\infty} \frac{1}{2\pi\sigma\sigma_s} e^{-\frac{x^2}{2\sigma_s{}^2}} e^{-\frac{(b-x)^2}{2\sigma^2}} db.dx \tag{3.20}$$

$$P = \frac{\tan^{-1}(\frac{\sigma}{\sigma_s})}{2\pi} \tag{3.21}$$

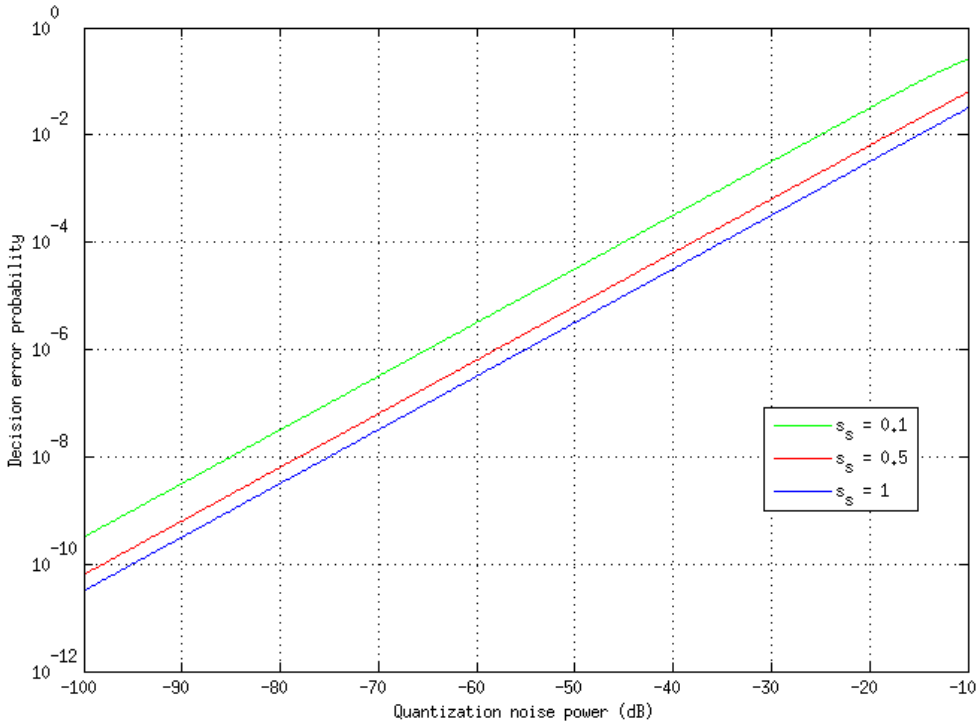


Figure 3.7 – Variation of total decision error probability with respect to the quantization noise power for different signal power $s_s = \sigma_s$.

Figure 3.7 shows the variation of the error probability P_e with respect to the quantization noise power $p_n(dB) = 10\log_{10}\sigma$ for three signals of different power. When the quantization noise deviation increases, the perturbation due to fixed-point conversion increases and the error probability at the output increases. When the power of the signal decreases the perturbation due to quantization noise has more effect at the output and leads to higher rate of decision error. If the quantization noise deviation is equal to 0.1, the error probability is 3% when the signal power is equal to 1. This probability decreases with the decrease of σ and becomes very low for a sufficiently low quantization noise power.

Decision operations are widely used in digital communication systems to detect the transmitted symbol. In [5], the authors evaluate the decision error on the 16-QAM and BPSK constellations. The error is detected by the comparison between the results in finite precision and in infinite precision. The source signals are generated such that the QAM symbols are uniformly distributed. The quantization noise and the channel noise are assumed to be a normal distribution with a zero mean and a standard deviation equals respectively to 1 and σ_q. Figure 3.8 shows the error rate versus the quantization noise power. When the quantization noise power decreases the error rate decreases for both demodulation schemes.

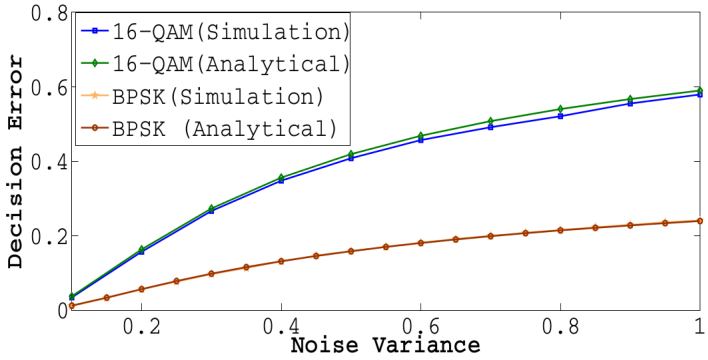


Figure 3.8 – Total decision error probability for 16-QAM and BPSK constellations from [5]

3.3 Selective simulation

The typical way to analyze the decision error and overflow effects on the application quality is to use fixed-point simulations. The advantage of such approaches is their capability to be performed on any kind of systems. Complete simulation-based approaches, which are often used in fixed-point optimization, lead to long execution times [6, 66]. They use specific data types to emulate the finite word-length effects on floating-point based machines and require a huge number of samples to obtain an accurate evaluation. Moreover, these approaches are exhaustive in the sense that all the input samples are simulated for each quality evaluation process. This is a great challenge for digital hardware design that aims to optimize the implementation cost and reduce the time-to-market.

The tool used to evaluate the application quality is a key stage in the world-length optimization process. In this context, we propose a new approach using selective simulations to accelerate the simulation of finite precision effect analysis. This approach can be applied on any application described in C language and based on repetitive processing on a set of input data. An index is associated with each input data. The index concept is detailed in section 3.3.1. The proposed approach simulates the application only when an overflow or an un-smooth error occurs. Indeed, errors must be rare events to maintain the system functionality. Selective simulation can reduce significantly the time required to evaluate the application quality criteria. Compared to analytical approaches, which are not trivial to be applied on all fixed-point systems, selective simulation approach maintains the advantage of simulation-based approaches by supporting any application described in C language. Compared to classical simulation approaches, which simulate all the input samples and lead to long execution time, selective simulation accelerates the evaluation of application quality. In the following, the proposed approach is described for the block level.

The decision flow diagram, presented in Figure 3.9, describes the decision making process. This decision tree is executed for each index of the input data. For each iteration of the optimization process a different word-length configuration is tested and a new evaluation of the quality is performed. In a given iteration, each operation is checked if it leads to an error for the tested word-length. If at least one operation encounters an error, then the system is simulated for the selected

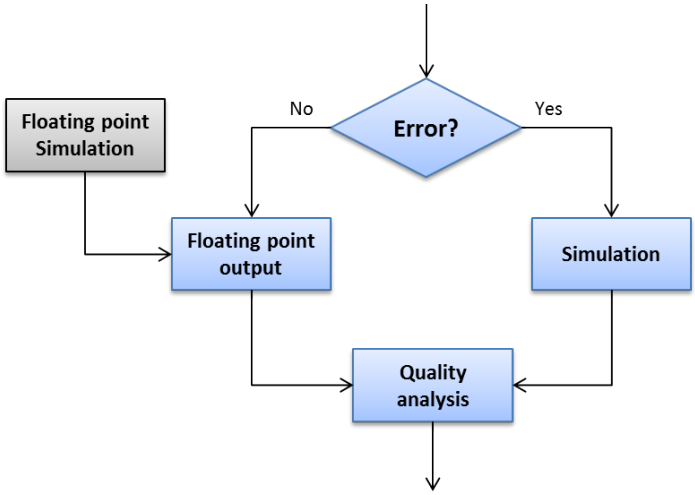


Figure 3.9 – Decision flow diagram for conditional simulation

index. If the output is not susceptible to generate an error, the output of the reference simulation is valid and there is no need to perform simulation.

The proposed approach for quality analysis associated with the fixed-point conversion flow is presented in Figure 3.12. The inputs of the proposed framework are the C source code of the considered application (SUT: System Under Test) and an associated testbench. This testbench includes the input stimuli provided by the system designer. He/she is in charge of selecting relevant stimuli to guarantee complete test coverage of the SUT. In the SUT C source code, pragmas can be used to specify the set of variables S_v, containing N_v variables for which their fixed-point conversion effects on the application quality are studied.

3.3.1 Index dimension

The representation and dimension(s) of the index depends on the studied application. In the following, we explain the effect of number of dimensions on the index format.

One dimension In digital signal processing systems, the input are sampled at equidistant time points, which corresponds to the sampling period. When the input signal x is sampled with a period T_e, the index n corresponds to the n^{th} sample as shown in Figure 3.10.

Two dimensions Two dimension indexes can be considered like in the case of matrices such as the pixel of an image. In this case, the index is represented by (i, j) where i and j indicate respectively the row and column.

Three dimensions The index can be represented as a three-dimensional object (k, i, j) as in the case of video processing applications, where k, i and j indicates respectively the frame number,

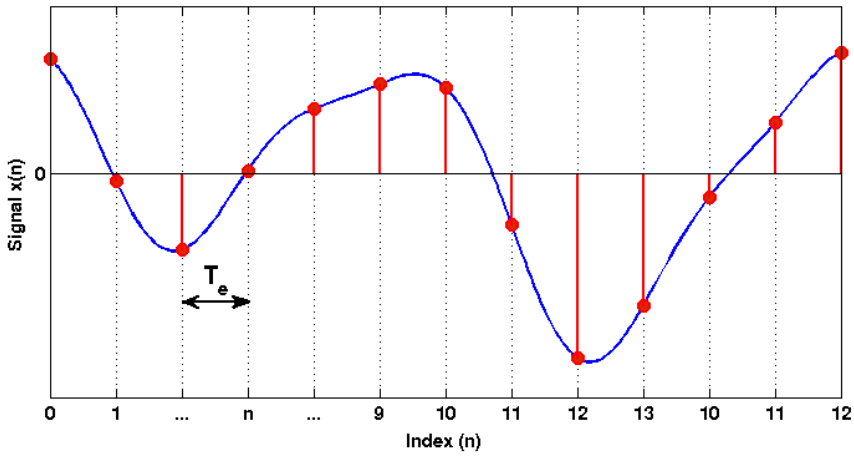


Figure 3.10 – One dimensional index of a sampled signal.

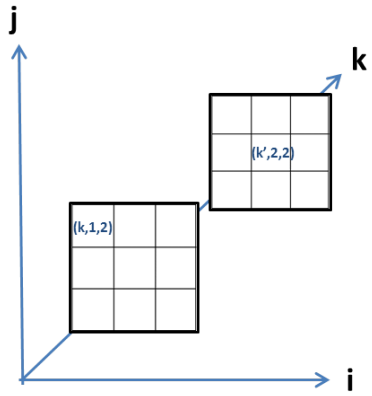


Figure 3.11 – Three dimensional index in video processing application.

the row and the column as shown in Figure 3.11.

3.3.2 Selective simulation technique

The proposed approach for selective simulation is made-up of four steps: *Code Instrumentation (T_0)*, *Index Classification (T_1)*, *Index Selection (T_2)* and *Selective Simulation (T_3)*. This approach is designed to be exploited by an iterative optimization algorithm for word-length refinement. This optimization process is made-up of N_v variables. At each iteration, the selective simulation technique is performed several times to evaluate the application quality.

T_0 and T_1 are applied as an introductory step for the optimization algorithm. These steps are executed only once and their results are used for different iterations of the optimization process. At each iteration, the system is evaluated N_v times to test the effect of the finite precision of each variable and T_2 and T_3 are executed in each quality evaluation. The four steps of this approach

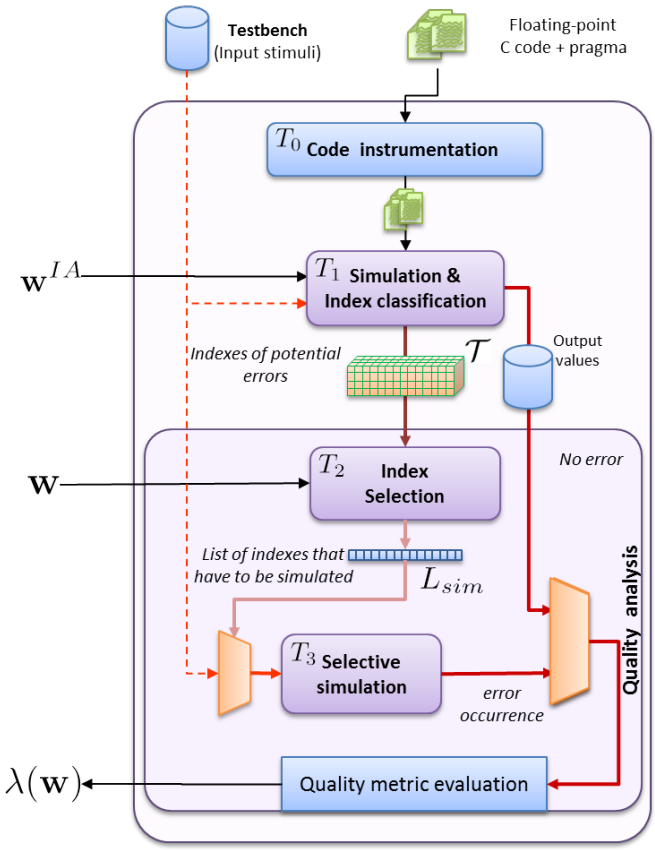


Figure 3.12 – Proposed framework for quality evaluation associated with fixed-point conversion.

are detailed in the following. The classical fixed-point simulations, where all the input indexes are simulated, are referred to as complete simulations. In the following, the term "error" refers to an overflow or un-smooth error.

Code instrumentation (T_0) The first step T_0 instruments the SUT C source code and generates different C++ files required by this module to analyze the finite precision effects. A similar instrumentation technique was used in [37] for dynamic range evaluation. The operator overloading concept associated with C++ object-oriented language is used to collect information and simulate finite precision effects. Currently this transformation is not automated, and the C source codes are modified manually, but the Gecos [87] source-to-source transformation tool can be used to carry-out this C source instrumentation step.

Index classification (T_1) The index classification step aims at identifying the indexes of potential errors for each variable x of the set S_v. It is carried-out by a complete simulation (floating-point) of the SUT system for the N_N input samples. The instrumented SUT C code is executed to process all the input samples and information are collected during the code execution. The indexes of the different values of x collected during the simulation are separated into N_{I_x} intervals depending on the number of bits of the integer part or the fractional part. N_{I_x} is a trade off between the amount

of data stored in the database and the size of the exploration space for the optimization process. For a variable x, member of the set S_v, the index n corresponds to a potential error if the following condition is satisfied

$$\left| v_x(n) - v_x^{limit} \right| > N_{I_x} \tag{3.22}$$

where $v_x(n)$ is the configuration tested for variable $x(n)$. it corresponds to IWL in the case of overflow occurrence and FWL in the case of un-smooth error. These criteria are detailed in chapter 4 for the case of overflow and in chapter 6 for un-smooth error. v_x^{limit} is the maximum value considered for $v_x(n)$ in the case of overflow and the minimum value considered for $v_x(n)$ in the case of un-smooth error.

The indexes of the potential errors are stored in the three dimensional structure $\mathcal{T}$. The size of $\mathcal{T}$ depends on N_v, the number of variables, N_{I_x}, the number of intervals considered for each variable x and the number of indexes to be stored for each interval of the variable x. The index n is stored in $\mathcal{T}_{x,j,k}$ where k indicates the rank of the index n in the interval I_j associated with the variable x considering the indexes of potential errors already recorded. The value of j is defined as follows

$$j(x(n)) = \left| v_x^{limit} - v_x(n) \right|. \tag{3.23}$$

Index selection (T_2) The second step aims at identifying the indexes which have to be simulated for a given configuration for each variable of the set S_v. Let $\mathbf{v}^{test}$ be the N_v-length vector containing the tested intervals for the different variables of the set S_v. For the case of overflow, $\mathbf{v}$ is computed from the word-length of the integer part of each variable of the set S_v as detailed in section 4.3.2. For the case of un-smooth errors, $\mathbf{v}$ is computed from the deviation of the quantization error associated to each variable of the set S_v as detailed in section 6.3.2.

The list of indexes L_{sim} that have to be simulated for the configuration $\mathbf{v}^{test}$ is obtained as follows

$$L_{sim}(\mathbf{v}) = \bigcup_{i=1}^{N_v} \Gamma_i \qquad \text{with } \Gamma_i = \begin{cases} \bigcup_{j=0}^{\Delta_{\mathbf{v}_i}} \mathcal{T}_{x_i,j} & \text{if } \Delta_{\mathbf{v}_i} > 0 \\ \phi & \text{otherwise} \end{cases} \tag{3.24}$$

$\Delta_{\mathbf{v}_i}$ represents the number of intervals that has to be simulated for analyzing the effects of the errors for the i^{th} variable when the configuration $\mathbf{v}_i^{test}$ is tested. $\Delta_{\mathbf{v}_i}$ is obtained with the following expression

$$\Delta_{\mathbf{v}_i} = |\mathbf{v}_i^{limit} - \mathbf{v}_i^{test}| \tag{3.25}$$

Selective simulation (T_3) This step determines the SUT output values when error occurs. The SUT is simulated for the indexes of L_{sim} only. No error occurs for indexes not included in L_{sim}.

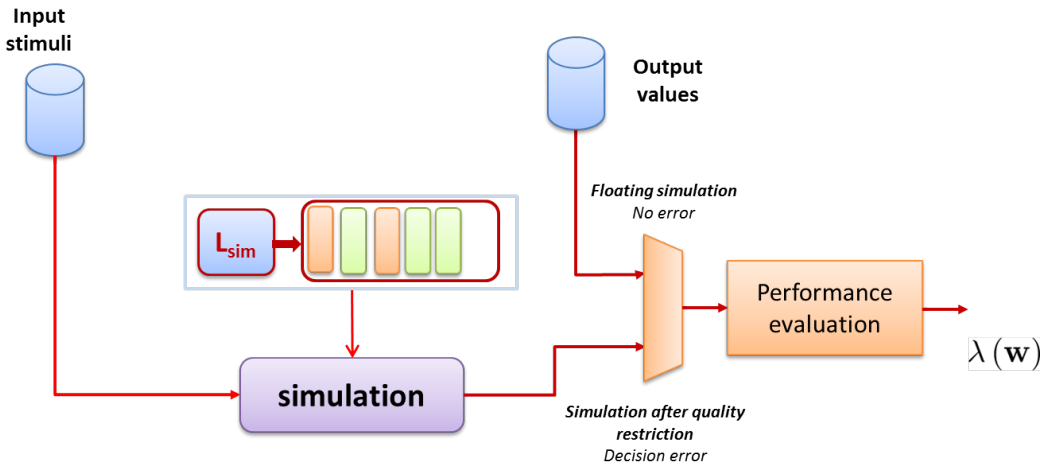


Figure 3.13 – Selective Simulation T_3

Thus no degradation on the application quality occurs and the output of the complete simulation done in step T_1 is used. The selective simulation step is illustrated in Figure 3.13.

When an error occurs, it may be necessary to simulate several consecutive indexes. If an error occurs at index n of an internal variable x, the SUT output y has to be computed for all the values of k for which $y(n+k)$ depends on $x(n)$ with $k \in \mathbb{N}$. In non-recursive systems, where none of the variables depends on its previous samples, the maximum value of k is equal to the number of delay operations along the path between x and y. In recursive systems, the output $y(n+k)$ has to be computed while the effect of $x(n)$ on $y(n+k)$ is not negligible. The maximum value of k depends on the length of the impulse response of the system having y as output and x as input. In the experiments, only non-recursive systems without delays have been considered.

3.3.3 Performance of the proposed approach

Our approach aims at accelerating the evaluation of error effects with respect to Complete Simulation Technique (*CST*) where all the N_N input samples are simulated. The time taken for the quality evaluation process is of interest when comparing the two techniques.

3.3.3.1 Complete simulation technique

A complete simulation is a fixed-point simulation of the N_N input samples. Let's consider that the quality metric is evaluated N_e times during the optimization process. The global execution time of the optimization process depends on the number of input samples N_N and the number of evaluation N_e. It can be expressed as

$$t^g_{cst} = N_e \cdot t_{cst} \tag{3.26}$$
$$= N_e \cdot N_N \cdot t_{sim} \tag{3.27}$$

where t_{cst} represents the execution time of one evaluation using complete simulation and t_{sim} represents the execution time taken by one sample simulation.

3.3.3.2 Selective simulation technique

In the proposed selective simulation technique, the application is only simulated for indexes in the list L_{sim}. The number of errors, and thus the size of list L_{sim}, varies from one evaluation to another depending on the tested configurations. The global execution time of the optimization process using the proposed approach t^g_{sst} is obtained with the following expression

$$t^g_{sst} = t_{s1} + \sum_{i=1}^{N_e} t^i_{sst} \tag{3.28}$$

where t_{s1} is the execution time of index classification T_1, t^i_{sst} represents the execution time of the i^{th} evaluation using selective simulation. t^i_{sst} depends on the number of occurrence of the error for the i^{th} evaluation and can be computed with the following expression

$$t^i_{sst} = t^i_{s2} + N^i_L.t_{sim} \tag{3.29}$$

where t^i_{s2} is the execution time of index selection T_2 and N^i_L is the number of indexes in the set L_{sim} for the i^{th} evaluation. In an optimization process, a high number of iterations is needed to converge into the final solution. Thus, the execution time t_{s1} of the first step becomes negligible and the global execution time ccan be computed with the following expression

$$t^g_{sst} \approx \sum_{i=1}^{N_e} t^i_{sst} \tag{3.30}$$

3.3.3.3 Accuracy of selective simulation technique

Selective simulation technique is designed to reduce the time of the application quality evaluation process. It is of high importance that the proposed technique maintains the accuracy of the complete simulation technique.

When the finite word-length does not generate any error, the outputs of the fixed-point simulation and the reference (floating-point) simulation are the same. The output of the floating-point simulation of all the input samples is stored in a database during step T_1. This output corresponds to the case where no error occurs. In step T_2, the indexes, for which an error occurs, are identified.

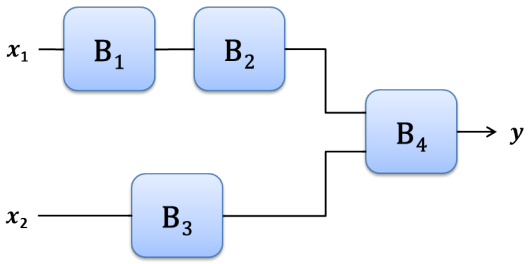


Figure 3.14 – System with multiple blocks

The proposed technique simulates the system with finite precision for these samples and use the output from the floating-point simulation database for the other samples.

For the complete simulation, a fixed-point simulation of the system under test is carried-out for all the input samples. In this case, the outputs for the samples without error are the same as the floating-point outputs, and the outputs for the samples with error occurrence are the same as the proposed technique. Thus, the proposed technique and complete simulation technique leads to the same quality metric evaluation.

3.4 System level evaluation

In the previous part, the selective simulation technique has been detailed and is able to handle a block of an application. In order to handle a complete application, a system level evaluation has to be considered. In a system decomposed into several blocks, errors at the output are due to different contributions of errors generated by different blocks. We consider a system S, as shown in figure 3.14. The system is decomposed into four blocks, with two inputs x_1 and x_2 and one output y. The errors at the output of B_1 are propagated through B_2. The errors at the output of B_2 and B_3 are propagated through B_4.

The errors at the output of each block B_i come from two sources as shown in Figure 3.15. The first source corresponds to the errors generated inside the block. The second source correspond to the errors generated inside previous blocks and propagated through the considered block. Let G be the function that determine the indexes of errors generated inside the block from the block variables word-length (w_{B_i}) Let P be the function that determines the indexes of the errors propagated from the previous blocks.

The proposed selective simulation technique is applicable on systems with more than one block. The index classification step is done for each block separately. Then, the index selection step determines the list of errors at the output of each block taking into account the errors generated inside the block and the propagated errors. For each block B_i, the set of indexes to be simulated L_i is the union of the list of indexes of errors L_i^g generated inside the block B_i and the set of indexes of errors L_i^{i-1} propagated from the previous block B_{i-1}. Each block B_i is simulated for the indexes in the list L_i during the selective simulation step.

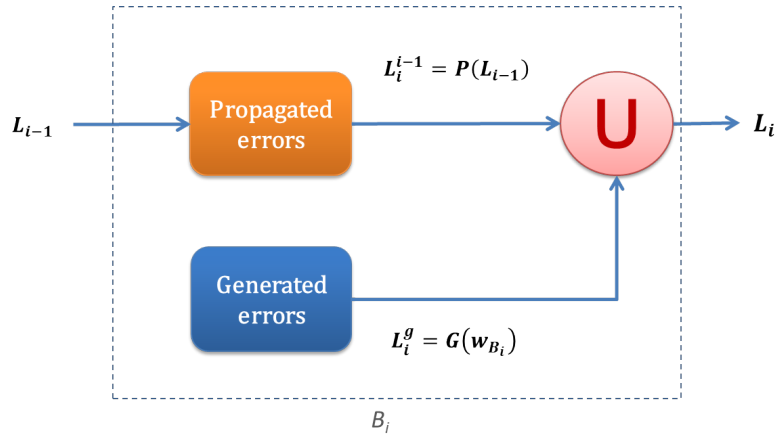


Figure 3.15 – Error propagation from one block to another.

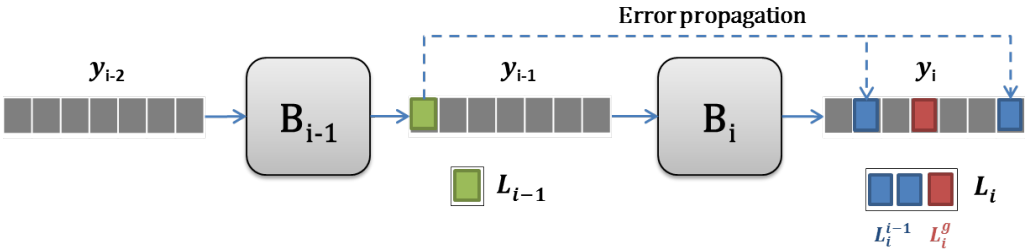


Figure 3.16 – Error generation and propagation through two consecutives block. The green indexes are the indexes of errors generated in B_{i-1}. The blue indexes are the indexes of errors propagated from $Bi-1$ to B_i. The red indexes are the indexes of errors generated in B_i.

Figure 3.16 shows the generation and the propagation of errors. The input y_{i-2} of block B_{i-1} is considered without errors. Let L_{i-1} be the set of indexes for which an error is generated at the output y_{i-1} of the block B_{i-1}. These errors are propagated through block B_i and give the set of indexes corresponding to the propagated errors L_i^{i-1}. In addition to L_i^{i-1}, the computation in B_i generates errors at indexes of the L_i^g. The set of indexes of the output of B_i corresponding to an error is $L_i = L_i^{i-1} \cup L_i^g$.

3.5 Conclusion

The problem of evaluating the application quality metric when an overflow or an un-smooth error occurs with low probability is addressed in this chapter. This chapter begins by explaining the rare event aspect of overflow and un-smooth errors. Overflow and un-smooth errors have high amplitude with respect to the signal amplitude and therefore a huge impact on the application quality metric. In the world-length optimization process, the quality metric is evaluated for each word-length configuration. However, not all input indexes lead to an error. Thus, the output of

most indexes does not change in comparison with reference simulation. Classical simulation approaches, which use complete simulation technique, simulate the system for all the input indexes. This imposes serious limitation in terms of design time.

This chapter introduces a new approach to accelerate the quality evaluation by selectively simulating the system at the indexes leading to errors. Thus, the effort for fixed-point simulation is greatly reduced. The proposed technique is performed with four steps. The first two steps are pre-processing steps, which are done only one time. The latest two select the indexes of errors and selectively simulate the system at these indexes. This will give the same results as those obtained with a classical simulation, but this approach reduce significantly the simulation time.

CHAPTER 4

Selective simulation in presence of overflow

4.1 Introduction

In the previous chapter, we presented the proposed selective simulation framework that aims at accelerating the evaluation of application quality. This chapter details the procedure and explains the framework specificities when applied in the presence of overflow. After presenting the methodology of variable selection for integer world length optimization, the implementation of the framework in $C++$ is explained when overflow occurrence is considered. Finally, experiments are conducted to verify the effectiveness of the proposed selective simulation technique.

4.1.1 Advanced fixed-point conversion flow

Figure 4.1 presents the advanced fixed-point process, where stages related to dynamic range evaluation and integer WL optimization are emphasized. The first stage corresponds to the dynamic range evaluation, which computes the minimal and maximal values of each variables. Interval arithmetic [39] or affine arithmetic [44] can be used to provide bounds which ensure no overflow. Otherwise, simulation based approach can be used as an alternative to obtain the minimal and maximal values of the studied variables. The second stage corresponds to the variable selection stage for Integer WL optimization. The aim of this stage is to select the variables for which the IWL will be optimized.

The third stage is the WL optimization stage, where the integer WL and the fractional WL are determined. The IWL optimization process requires to evaluate the effect of overflow on the application quality metric. The execution time of application quality evaluation significantly affects the duration of the optimization process. In this context, the proposed selective simulation technique is designed to be exploited by an IWL optimization strategy. In contrast to classical simulation approaches, where all the samples are simulated, the decision to perform fixed-point simulation to evaluate the system is subject to the signal conditions in the proposed selective simulation approach. This leads to lower number of evaluations, and thus accelerate the process of IWL optimization.

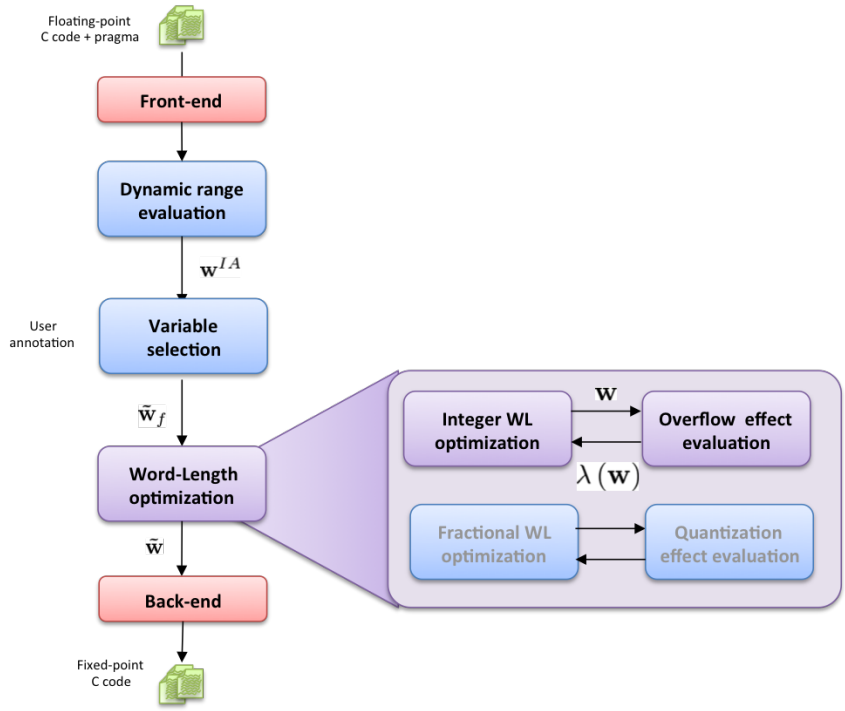


Figure 4.1 – Advanced fixed-point process integrating integer word-length

The input of our fixed-point conversion flow is a C code. To limit the modification of the original floating-point C program, the operator overloading concept of C++ language is used. New data classes for tracing the range of the signal or to simulate overflow mechanism are proposed and the possible overflow is named as `SigOvf`. Thus, it is only necessary to change the type of the considered variables from $float$ to the new C++ class.

4.1.2 Illustration with an example

To illustrate the different stages of the advanced fixed-point conversion flow presented in figure 4.1, the example presented in Listing 4.1 is used. This simple C code to compute a 3^{rd}-order polynomial. The input x and the output y are two signals of $N = 100000$ indexes such that $y = x + 0.333x^3$. In the following sections, the application of the proposed class on the code to accelerate the simulation is shown.

4.2 Variable selection for integer word-length optimization

The aim of this stage is to select the variables for which the IWL will be optimized during the WL optimization process. Given that overflow may significantly degrades the application quality metric, the overflow probability must be kept in reasonable bounds. Consider a variable having an integer WL that ensures no overflow equal to w_{IA}. If reducing the integer WL of this

```
#define  N  100000
float Xin[N] = {..,..  };
float Yout[N];
void ComputePoly(){
   float Ytmp;
   float Coeff1, Coeff2;
   Coeff1 = 1;
   Coeff2 = 0.333;
   for(int indice = 0 ; indice < N; indice ++) {
      Ytmp = Coeff1+Coeff2*Xin[i]*Xin[i];
      Yout[i] = Xin[i]*Ytmp;
     }
```

Listing 4.1 – Floating-point C code to compute a 3^{rd}-order polynomial

variable by a single bit significantly increases the overflow probability and degrades unacceptably the application quality, then it is a waste of time to consider the variable in the optimization process.

Determining whether IWL reduction is worth considering for a variable x is decided by analyzing the overflow probability, P_{ovf}. Let $P_{ovf}(w)$ be the overflow probability of a variable x for which the integer word-length is set to w. In the case of signed number, the expression of the overflow probability $P_{ovf}(w)$ is as follows

$$P_{ovf}(w) = \int_{-\infty}^{-2^w} f_x(x)dx + \int_{2^w-1}^{+\infty} f_x(x)dx, \tag{4.1}$$

where $f_x(x)$ is the probability density function of variable x. In the case of unsigned number, the expression of the overflow probability $P_{ovf}(w)$ is as follows

$$P_{ovf}(w) = \int_{2^{w+1}-1}^{+\infty} f_x(x)dx, \tag{4.2}$$

The determination of the overflow probability requires the knowledge of the probability density function $f_x(x)$ or more specifically the tail of the probability density function. As detailled in Section 2.2.1.2, stochastic approaches can be used to determine the probability density function (*PDF*) of the data from the system input statistical characteristics [18, 19]. Other techniques use Extreme Values Theory to determine the data PDF tail [15, 17].

The discrete probability density function $f_w(w_x)$ of w_x can be also used to compute the overflow probability $P_{ovf}(w)$. The term w_x corresponds to the integer WL required to represent the value x. By considering $f_w(w_x)$, the expression of the overflow probability $P_{ovf}(w)$ is as follows

$$P_{ovf}(w) = \sum_{w_x=w}^{\infty} f_w(w_x). \tag{4.3}$$

```
class histog {
private:
   std::string name;     // Variable name
   int valIWL[Nelt];     // Table of occurrence of each iwl.
   int countIWL;            // Total number of iwl occurrences
   int wIA;              // IWL required to avoid overflow
public:
      // class methods
   ...
};
```

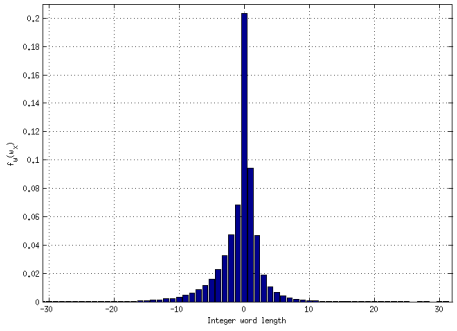
Listing 4.2 – Definition of histog C++ class

```
histog & histog::operator = (float v)
int index;
  index = compteIWL(x) + Nelt - wIA;
  if(index < 0) index = 0;
  valIWL[index]++;
  countIWL++;
  return *this;
}
```

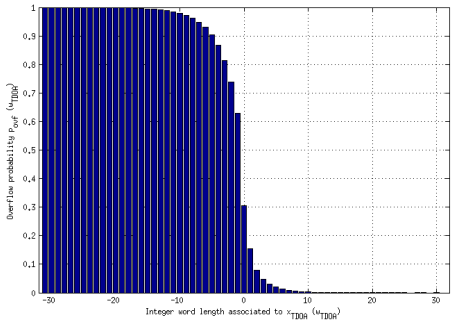
Listing 4.3 – Overloading of the affectation operation for histog C++ class

As an alternative to analytical approaches that computes $f_x(x)$, we propose a simulation-based method that estimates the PDF $f_w(w_x)$ by analyzing the input data set. The main advantage of the proposed approach is its implementation simplicity. Although the results of the proposed method depends on the input data set, selecting a suitable data set that reflects the real input overcomes this challenge. Given that $w(x)$ are integer values, the probability density function $f_w(w_x)$ can be computed easily from data collected during simulation. To collect the data during simulation, a C++ class `histog` is defined as presented in Listing 4.2 and the data are collected when a value is affected to a `histog` object. For each value of the variable x, the integer WL $w(x)$ is computed and the bin associated to this value $w(x)$ is incremented as presented in Algorithm 4.3. A similar technique is proposed in the Matlab fixed-point designer tool [8].

To illustrate the results obtained with this C++ class, two variables are considered. The first one is a variable x_{TDOA} used in the computation of mobile position in a Global Position System. For the considered variable the dynamic range is 32 bits. Figure 4.2(a) shows the PDF $f_w(w_x)$ of the variable x_{TDOA}. It has a long-tailed distribution, which gives the possibility to reduce the IWL without huge loss in the quality. This is illustrated in Figure 4.2(b) that shows the probability of overflow in terms of the integer word length w_{TDOA} allocated to the variable x_{TDOA}. Reducing the integer WL by 10 bits leads to overflow probability equal to 0.3%. This probability increases to

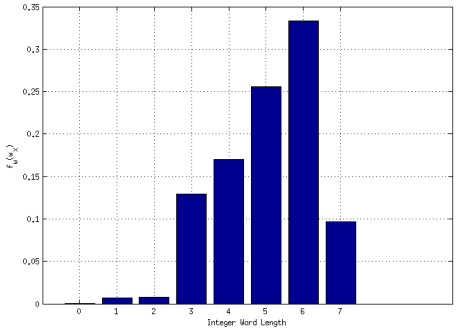


(a) Discrete probability density function $f_w(w_x)$ of the integer WL

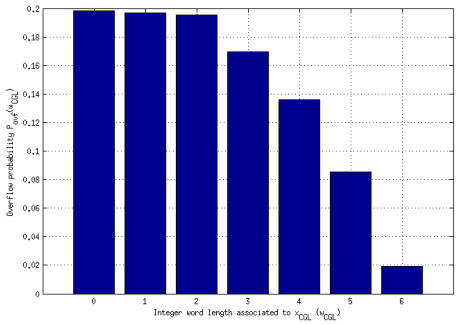


(b) Overlow probability $P_{ovf}(w)$

Figure 4.2 – A variable x_{TDOA} used in GPS system



(a) Discrete probability density function $f_w(w_x)$ of the integer WL



(b) Overlow probability $P_{ovf}(w)$

Figure 4.3 – A variable x_{CGL} used in an image processing system

5% with the reduction of 17 bits. Thus, this variable is considered as a good candidate for integer WL reduction.

The second variable is the output image x_{CGL} of a block computing the grey image from an input RGB image. Figure 4.3(a) shows the PDF $f_w(w_x)$ of of the variable x_{CGL}. The distribution has short-tail, which does not give freedom to reduce the number of bits of the integer part. Reducing the integer part comes at the expense of a high quality degradation. For example, reducing of 1 bits gives an overflow probability of 2% while reducing of 2 bits increases the overflow probability to 8.5%. Thus, this variable is not a good candidate to be selected for the study of integer word length optimization. The margin of reducing the number of bits of the integer part is limited to 1 bit in this example. Otherwise, the quality degradation does not respect the required quality constraint.

In order to automate the selection of the variables considered in the optimization process, we evaluate the variable in terms of IWL reduction. For a variable x, we define $\Delta_w(P)$ as the difference between w_{IA}, the integer WL that guarantees no overflow, and w_P, the minimal number

of bits that can be used while the probability of overflow $P_{ovf}(w_P)$ lower than P.

$$\Delta_w(P) = w_{IA} - w_P, \tag{4.4}$$

with

$$w_P = \min(w) \quad \text{such as} P_{ovf}(w) < P \tag{4.5}$$

A variable x is considered as a good candidate for IWL reduction if $\Delta_w(P)$ is greater than a value $\Delta_w^{\min}$ for a maximal probability of overflow of $P_{ovf}^{\max}$. The values $\Delta_w^{\min}$ and $P_{ovf}^{\max}$ depend on the application. They are defined by the user and classical values are around 2 bits for $\Delta_w^{\min}$ and 1% for $P_{ovf}^{\max}$.

4.3 Quality Evaluation in the Presence of Overflow

In this section, the specificities of the selective simulation framework, presented in chapter 3, are presented in the case of overflow analysis. Subsequently, the steps T_1 to T_3 of the selective simulation technique are detailed in the context of overflow analysis.

4.3.1 Index classification (T_1)

The aim of the Index Classification step is to identify the indexes of potential overflows for each variable x of the set S_v, which is the set of considered variables. These indexes will be used in step T_3 to selectively simulate the system. The value $\mathbf{w}_x(n)$ corresponds to the IWL of the variable x at index n. The index n corresponds to a potential overflow if the following condition is satisfied

$$\mathbf{w}_x(n) > \mathbf{w}_x^{max} - N_{I_x}, \tag{4.6}$$

where N_{I_x} is the number of interval recorded for the variable x. $\mathbf{w}_x^{max}$ is the maximum number of bits, recorded through simulation of all indexes, needed to represent the integer part of x. It must be noted that $\mathbf{w}_x^{max} \leq \mathbf{w}_x^{IA}$, where $\mathbf{w}_x^{IA}$ is the IWL obtained using interval arithmetic. The IWL of the variable x at index n, $\mathbf{w}_x(n)$, represents the number of bits required to code the integer part of the value $x(n)$ and is computed as:

$$\mathbf{w}_x(n) = \lceil \log_2(|x(n)|) \rceil + \alpha \tag{4.7}$$

$$\alpha = \begin{cases} 2 & \text{for } mod(\log_2(x(n)), 1) = 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.8}$$

```
class SigOvf{
private:
  std::string  name;     // Variable name
  double     val;      // Floating-point value
  int        iwlMax;      // Maximal IWL considered
  int        iwlMin;         // Minimal IWL considered
  unsigned int listSizeMax;    // Maximal number of element in the list
  std::vector<ListIndex *> vectListIndex; // vector of list of instant
public:
   ... // Class methods
```

Listing 4.4 – Definition of SigOvf C++ class

The index n of the potential overflows is stored in the structure $\mathcal{T}_{x,j,k}$, where k indicates the rank of the index n in the interval I_j associated with the variable x considering the indexes of potential errors already recorded. The value of j is defined as follows:

$$j(x(n)) = \mathbf{w}_x^{max} - \mathbf{w}_x(n) \tag{4.9}$$

4.3.1.1 Implementation with SigOvf C++ class

The variable range is collected during the simulation of the `SigOvf` C++ class. This class computes the range of the variable at the current index and evaluates the corresponding number of bit. Then, the class searches in the structure $\mathcal{T}$ its suitable rank.

The class has several private members, as shown in the Listing 4.4. The variables $iwlMax$ and $iwlMin$ specify the limits of IWL for which overflow effect is analyzed. This will also allow limiting the size of structure T. The term $listSizeMax$ is the maximum number of index to be stored in the list of each vector of T. The term $vectorSize$ is the total number of list of T. The term val is the floating value of the variable.

The class `SigOvf` overloads arithmetic and relational operators. Hence, basic arithmetic operations such as addition, subtraction, multiplication, and division are conducted automatically for `SigOvf` variables. This property is also applicable for relational operators (" $==$ "," $!=$ "," $>$ "...). Therefore, any `SigOvf` variable can be compared to floating-point variables and constants. The contents, or private members, of a variable declared by the `SigOvf` class are updated when the variable is assigned by one of the assignment operators. The example of the affectation operation is presented in Listing 4.5.

```
SigOvf& SigOvf::operator = (double d){
   int iwl;
   val = d;
   std::vector<ListIndex *>::iterator itVect;
      iwl = compteIWL(d);
   if((iwl >= iwlMin)&&(iwl <= iwlMax)){
      if(vectListIndex[iwl - iwlMin]->size() < listSizeMax){
         vectListIndex[iwl - iwlMin]->push_back(ptrOvfContext->getIndex());
      }
   }
   ...
  return *this;
}
```

Listing 4.5 – Part of the code for overloading of the affectation operation for SigOvf C++ class

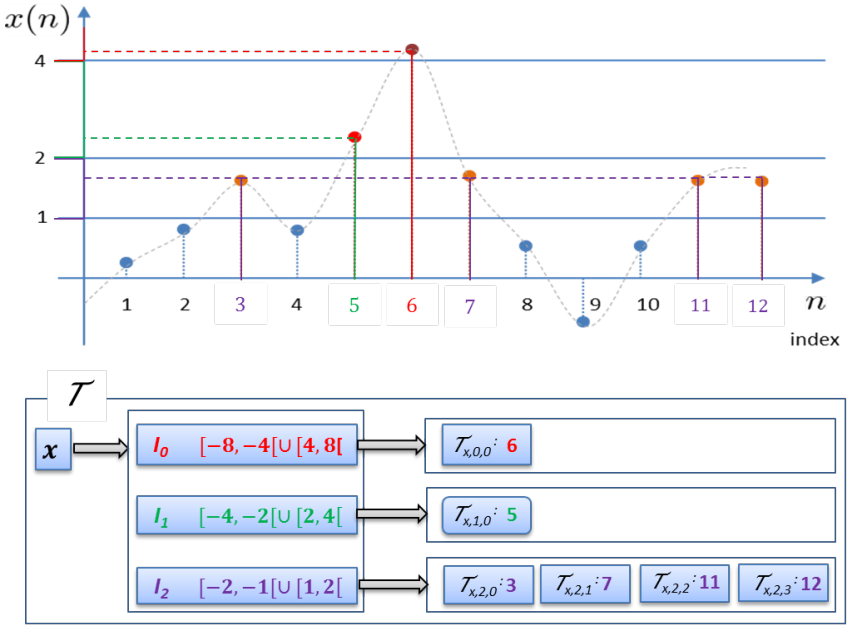


Figure 4.4 – Example of variable x evolution.

4.3.1.2 Illustration on the 3^{rd} polynomial example

Let consider the 3^{rd} polynomial example presented in Section 4.1.2 and aninput variable x having an IWL computed using interval arithmetic $\mathbf{w}_x^{IA}$ equal to 4. The values of x for the first indexes 0 to 12 are presented in Figure 4.4. In this example, $\mathbf{w}_x^{max}$ recorded as far is 4 bits ($\mathbf{w}_x(6)$). If N_{I_x} is chosen to be 3, the structure $\mathcal{T}$ at index 12 of the variable x, shown in the Figure 4.4, is as follows:

```
1  void ComputePoly()
2  {
3
4     SigOvf Ytmp(&OvfCtxt,"Ytmp");
5     SigOvf Yout(&OvfCtxt,"Yout");
6
7     float Coeff1, Coeff2;
8     Coeff1 = 1;
9     Coeff2 = 1/3;
10
11    /***** Scenario Detection ****/
12
13    for(int indice = 0 ; indice < 1000; indice ++)
14      {
15       Ytmp[indice] = Coeff1+Coeff2*Xin[indice]*Xin[indice];
16       Yout[indice] = Xin[indice]*Ytmp[indice];
17
18       OvfCtxt.incrementInstant();
19      }
20
21 }
```

Listing 4.6 – Code to collect informations for the steps T1 and T2 and for the 3^{rd} polynomial example

$$\begin{aligned} \mathcal{T}_{x,0} &= \{6\} && \text{with } I_0 = [-8,-4[\cup[4,8[\\ \mathcal{T}_{x,1} &= \{5\} && \text{with } I_1 = [-4,-2[\cup[2,4[\\ \mathcal{T}_{x,2} &= \{3,7,11,12\} && \text{with } I_2 = [-2,-1[\cup[1,2[\end{aligned} \tag{4.10}$$

At index 3, $x(3) = 1.6$ which gives $\mathbf{w}_x(3) = 2$ and $j(x(3)) = 2$. Thus index $3 \in I_2$. The listing 4.6 shows the modified 3^{rd} polynomial code to apply index classification. Each studied variable is redifined using the type $SigOvf$, each index is classified and saved in $\mathcal{T}$ at the affectation operator if the condition in Equation 4.6 is satisfied. $OvfCtxt$ determines the size of $\mathcal{T}$ and the index dimensions.

4.3.2 Index selection (T_2)

The Index selection step aims at identifying the indexes that have to be simulated for a tested configuration (vector) of IWLs. Let $\mathbf{w}$ be the N_v-length vector containing the tested IWLs of the

different variables of the set S_v, $x_i \in S_v$. The list of indexes for the configuration $\mathbf{w}$, $L_{sim}(\mathbf{w})$, that have to be simulated is obtained as follows:

$$L_{sim}(\mathbf{w}) = \bigcup_{i=1}^{N_v} \Gamma_i \qquad \text{with } \Gamma_i = \begin{cases} \bigcup_{j=0}^{\Delta_{w_{x_i}}-1} \mathcal{T}_{x_i,j} & \text{if } \Delta_{w_{x_i}} > 0 \\ \phi & \text{otherwise} \end{cases} \tag{4.11}$$

where $\Delta_{w_{x_i}} = \mathbf{w}_{x_i}^{max} - \mathbf{w}_{x_i}^{test}$. $\Delta_{w_{x_i}}$ represents the number of intervals that has to be simulated for analyzing the effects of overflow for variable x_i with $\mathbf{w}_{x_i}^{test}$ as tested IWL. If $\mathbf{w}_{x_i}^{test} = \mathbf{w}_{x_i}^{max}$ and $\forall x_i \in S_V \mid \exists\ \mathbf{w}_{x_i}^{max} < \mathbf{w}_{x_i}^{IA}$, then the implementation cost can be reduced without any degradation of the application quality.

For the example presented in Figure 4.4, $L_{sim} = \{6\}$ if the tested IWL $\mathbf{w}_x^{test} = \mathbf{w}_x^{max} = 4$, and $L_{sim} = \{5, 6\}$ if $\mathbf{w}_x^{test} = 3$.

In Listing 4.9, OvfList contains all the indexes leading to overflow according to the integer word-length $w1$ and $w2$ allocated to the two studied variables $Ytmp_s$ and $Yout_s$ respectively.

4.3.3 Selective simulation (T_3)

In the proposed selective simulation technique, the application is only simulated for indexes in the list L_{sim} corresponding to overflow occurrence. Two types of data are used to simulate the effect of overflow. Traditional fixed-point data types can be used to simulate the limitation of the number of bits for the integer and fractional parts and allow obtaining bit-accurate results. The proposed class data types are used for fast execution. The operations associated with the fixed-point data class, such as $=$, $+$, $-$, $*$ and $/$ are also defined at the class declaration. Then, fixed-point arithmetic operations, instead of floating-point arithmetic, are conducted automatically due to the operator overloading capability of$C++$.

A new data class $SimOvf$ have been developed to simulate the effect of overflow more quickly than traditional fixed-point data types. The declaration part of the $SimOvf$ class is given in the listing 4.7

As shown in the code, the class $SimOvf$ has several private members, which are the IWL associated to the variable, the minimal and the maximal values that can be represented according to the IWL, its name, the floating value of the variable and the selected overflow mode (saturation, round). The class supports all of the assignment and arithmetic operations. The assignment operator $=$ converts the input data according to the IWL of the left side variable and assigns the converted data to this variable. The input data, which is the evaluated result at the right side, can be either a floating-point or a $SimOvf$ data types. If the given IWL of the left side variable does not provide enough range for representing the input data, the data is modified according to the fixed-point attributes of the left side variable, such as saturation or wrap-around. The overloading of the assignment operator $=$ for the $SimOvf$ class is given in the listing 4.8.

For the example presented in Listing 4.1, the modified code to selectively simulate the 3^{rd}

```
class SimOvf{
private:
   string Name;          // Name of the variable
   int IWL;           // Integer word-length
   float    valMAX          // maximal value that can be represented
   float    valMIN          // minimal value that can be represented
   enum ModeOvfType modeOvf; //overflow mode saturation (sat) or wrap-around (wpa)

public:
   double value;
   ...    // Class methods
};
```

Listing 4.7 – Definition of the SimOvf C++ class

polynomial example is given in Listing 4.9.

4.4 Experiment and results

Experiments have been conducted on two applications corresponding to a Global Positioning System (*GPS*) and an Orthogonal Frequency-Division Multiplexing (*OFDM*) transmission chain. For both applications, the quality of the selective simulation approach is evaluated. For each application, the outputs of the selective simulation and the complete simulation techniques have been compared to verify the quality of the proposed approach. Throughout the experiments, both approaches lead rigorously to the same outputs, which verifies the accuracy of our approach to analyze overflow effect.

4.4.1 Global Positioning System

4.4.1.1 Application description

The first application is a Global Positioning System (*GPS*). The application, implemented in C++, aims at obtaining the three-dimensional position of a mobile given the locations of four GPS satellites and their Signal Time of Arrival (*TOA*s). The technique used to geolocalise the mobile user is the hyperbolic position location technique, also known as the Time Difference of Arrival (*TDOA*) position location method [88].

Figure 4.5 shows the main stages for determining the mobile position. The input signals of the system are the vectors x, y and z representing the positions of the four GPS satellites i, j, k and l in meters and the vector TOA representing the time of arrival of signals sent by each satellite in

```
SimOvf& SimOvf::operator = (double v){
  if(v >= valMAX){
    if(modeOvf==sat){
     value = valMAX;
    }
    if(modeOvf==wpa){   // wrap-around overflow mode
       value= fmod(v,(valMAX))-valMAX; // modulo(v, valMAX);
    }
  }
  else{
  if(v<=valMIN)   {
      if(modeOvf==sat)         {
         value=valMIN;
      }
      if(modeOvf==wpa){
           value= fmod(v,(valMIN))-valMIN;//modulo(v, valMIN);
      }
  }
  else   {
    value=v;
  }
 }
}
```

Listing 4.8 – Code for overloading of the assignment operator

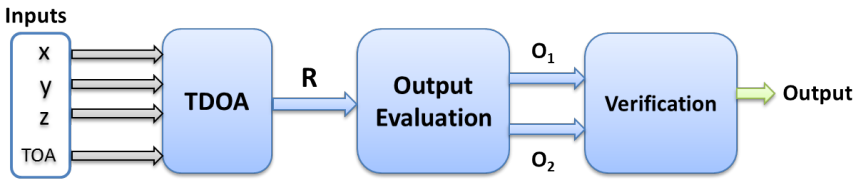


Figure 4.5 – Synoptic of a GPS using TDOA technique.

nanoseconds.

The first stage calculates the TDOAs R_{ij}, R_{ik}, R_{kj} and R_{kl} forming the vector R. The second stage evaluates two output signals o_1 and o_2 containing the two possible coordinates of the mobile position. The correct position is chosen by calculating the TDOAs of each output position in the last stage. The details of the algorithm can be found in [89].

The overflow effect is studied on the inputs of the system, the outputs of the 3 stages and the intermediate variables used in stage 2 and 3. 66 variables have been studied. Experiments have been conducted for 100000 different mobile positions.

```
1
2  #define  N  100000
3  float Xin[N] = {.,.,. };
4
5  void ComputePoly(){
6     SimOvf Ytmp_s(&OvfCtxt,w2,'s');      // w2 : Integer Word-length of Ytmp
7     SimOvf Yout_s(&OvfCtxt,w3,'s');      // w3 : Integer Word-length of Yout
8     float Coeff1, Coeff2;
9     Coeff1 = 1;
10    Coeff2 = 0.333;
11
12
13     /***** Index Selection ****/
14
15     OvfList =  OvfCtxt.SelectionIndex();
16
17    /***** Selective Simulation ****/
18     for(it = OvfList.begin() ; it!=OvfList.end(); it ++){
19      Ytmp_s[*it] = Coeff1+Coeff2*Xin[*it]*Xin[*it];
20      Yout_s[*it] = Xin[*it]*Ytmp_s[*it];
21     }
22 }
```

Listing 4.9 – Code for selective simulation (step T3) for the 3^{rd} polynomial example

Floating-point simulation is used as the reference to compute the fixed-point quality degradation. Floating-point simulation is a one time effort that can be considered as a pre-processing step. The quality degradation is evaluated through the normalized distance error (δ_d) between the positions computed by floating and fixed simulation. δ_d is calculated as follows:

$$\delta_d = \frac{\sqrt{\left(x_E^{ovf} - x_E^{ref}\right)^2 + \left(y_E^{ovf} - y_E^{ref}\right)^2 + \left(z_E^{ovf} - z_E^{ref}\right)^2}}{\sqrt{\left(x_E^{ref}\right)^2 + \left(y_E^{ref}\right)^2 + \left(z_E^{ref}\right)^2}} \tag{4.12}$$

where $(x_E^{ref}, y_E^{ref}, z_E^{ref})$ are the reference coordinates of the mobile positions and $(x_E^{ovf}, y_E^{ovf}, z_E^{ovf})$ are the coordinates of the mobile positions computed when overflows are considered.

4.4.1.2 Time improvement

Figure 4.6 shows the durations t_{cst} and t_{sst}^i, defined in Equations 3.26 and 3.29, with respect to the overflow probability. The execution time of the position evaluations using complete simulation technique, t_{cst}, is equal to 4.5 s independently of the overflow occurrence. Evidently, selective

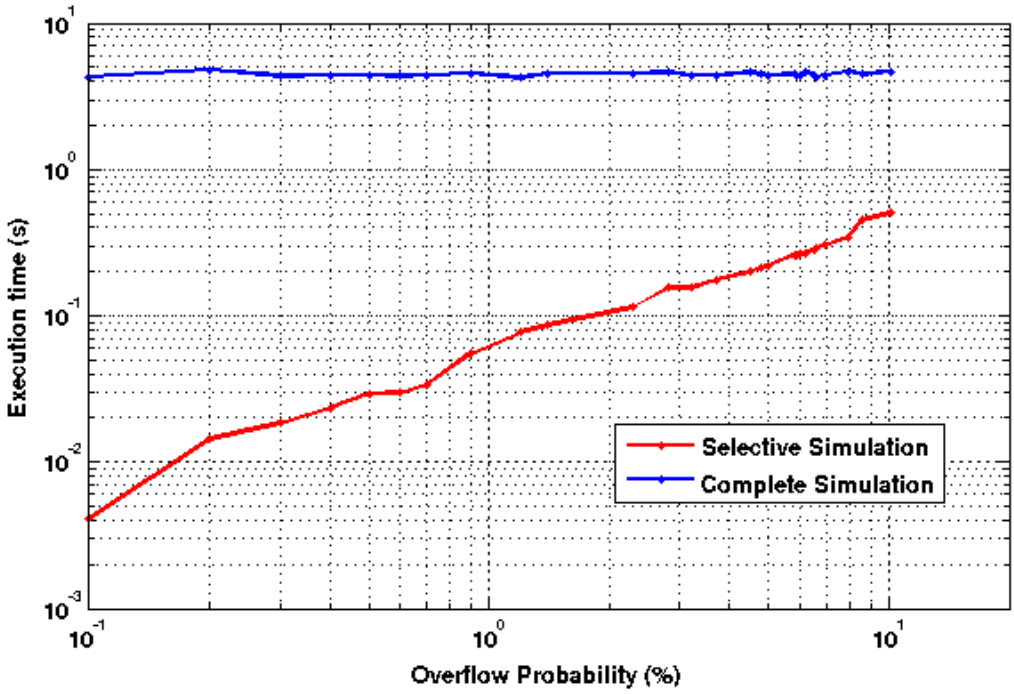


Figure 4.6 – Execution time of GPS application according to overflow probability.

simulation technique achieves lower execution durations depending on the overflow probability. As the overflow probability decreases, the improvement in time increases to reach a gain of up to 1200 times.

4.4.2 Orthogonal Frequency-Division Multiplexing Transmission Chain

4.4.2.1 Application description

The second application is an orthogonal frequency-division multiplexing (*OFDM*) transmission chain. Due to the high spectral efficiency of OFDM, it is now used in most of the new wired and wireless communications such as LTE, WiFi, digital television and optical communication [90]. It is an efficient mechanism for transmitting data over frequency-selective fading channels, where the channel division makes it possible to avoid difficult equalization schemes at the receiver.

The synoptic of an OFDM transmission system is presented in Figure 4.7. The application is based on an OFDM modulation with 64 subcarriers, each one being modulated with a BPSK scheme. From the total of 64 subcarriers, only 52 subcarriers are actually used for the transmission. The transmitter modulates the input bitstream and then sends it through a noisy channel. The transmitter is made-up of a mapper, an Inverse Fast Fourier Transform (*IFFT*) and the cyclic prefix insertion module. The mapper generates the BPSK constellation for each sub-carrier. The main block of the transmitter is formed by the 64-point IFFT. Then, a cyclic prefix with a length of 16 is used in order to avoid the inter-symbol interference. In this experiment, the channel model

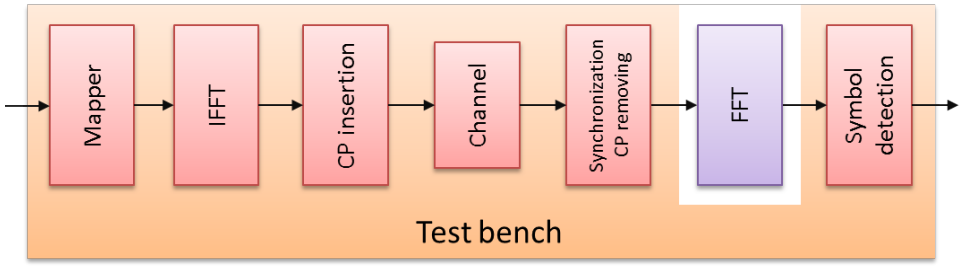


Figure 4.7 – Synoptic of an OFDM transmission system.

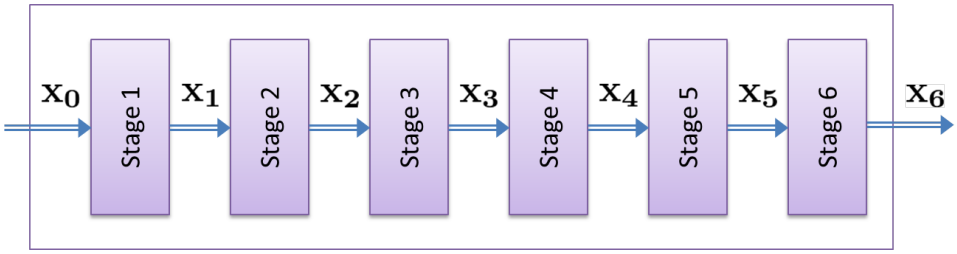


Figure 4.8 – Radix-2 64-point FFT with 6 stages.

corresponds to an additive white Gaussian noise. The receiver is made-up of the synchronization and cyclic prefix removing module, a FFT and the symbol detection. The first module carries-out the different synchronizations and removes the cyclic prefix, where a perfect synchronization is considered in our case. Then, the FFT is used to transform the complex time-domain samples into the frequency domain. For the BPSK modulation, the symbol detection is carried-out by analyzing the sign of the real part of each FFT output channel. Then, a parallel to serial transformation allows obtaining the bitstream.

The proposed approach is applied on the FFT part, which is a high computational part and the most challenging module in the receiver. The radix-2 64-point FFT is implemented with 6 stages carrying-out the butterfly operations (Figure 4.8). The effect of overflows is considered on the input of the system and the outputs of the 6 stages, i.e. 7 variables to be studied. The real part and the complex part of these variables have the same IWL. Experiments have been conducted with both complete simulation technique and the proposed selective simulation based approach for a number of indexes $N_N = 150000$. The bit error rate (*BER*) is the criteria used for evaluating the application quality. The quality degradation due to overflow is evaluated through the BER degradation Δ_{BER} defined as follows:

$$\Delta_{BER} = \frac{BER_{ovf} - BER_{ref}}{BER_{ref}} \tag{4.13}$$

where BER_{ovf} and BER_{ref} are the BER obtained with and without overflows respectively. The BER_{ref} is the BER obtained in the case of the IWL corresponds to the IWL computed with interval arithmetic ($\mathbf{w}^{IA}$). In these experiments, $\mathbf{w}^{IA} = [3, 4, 5, 6, 7, 8, 9]$

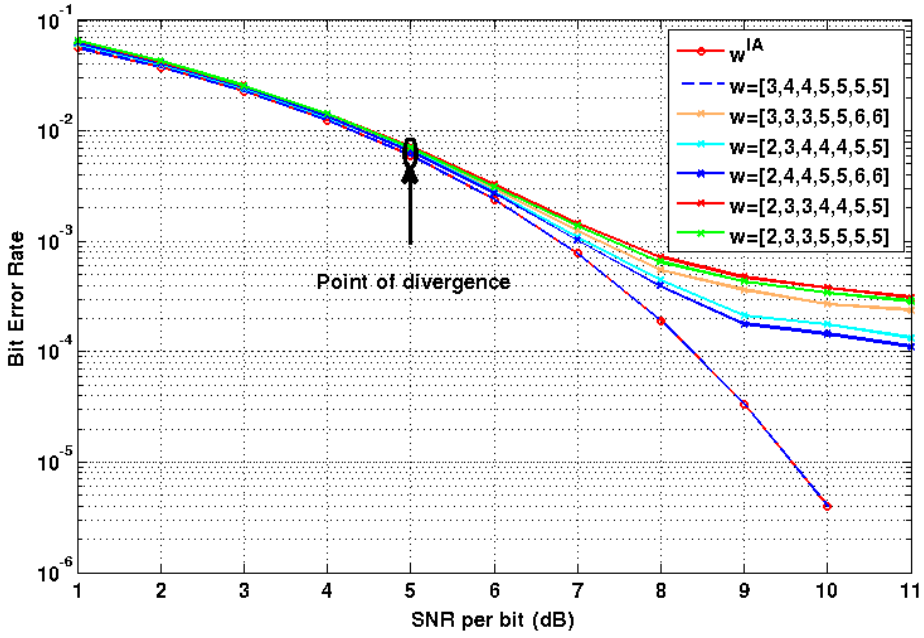


Figure 4.9 – BER vs SNR per bit variation for different IWL configurations.

4.4.3 Fast overflow effect analysis

4.4.3.1 Quality criteria analysis

Figure 4.9 shows the BER evolution for different IWL configurations according to the signal to noise ratio (*SNR*) per bit. $\mathbf{w}^{IA}$ is considered as a reference case corresponding to no overflow occurrence. For the IWL configuration $[3, 4, 4, 5, 5, 5, 5,]$, the BER curve overlaps the case of no overflow occurrence. This IWL decreases the cost without any quality degradation, and verifies that interval arithmetic may over-estimate the IWL. For other configurations, the BER curves are very close to the one obtained with no overflow occurrence at low SNR per bit. However, the differences between the curves start to appear for a specific SNR per bit. This SNR per bit is defined as "point of divergence". In the studied experiments, the point of divergence is 5 dB. For SNRs per bit greater than the point of divergence, the effect of overflow occurrences on the application quality criteria (BER) starts to appear. In this interval of SNRs per bit, the trade off between the cost, due to IWL configuration, and the application quality degradation, due to overflow probability, can be studied.

For high SNR per bit, the BER due to transmission noise is negligible with respect to the BER caused by overflow. The curve obtained in the case of no overflow occurrence tends to a zero BER when the noise power is very small, while the quality curves of other configurations flatten exhibiting the error-floor phenomenon.

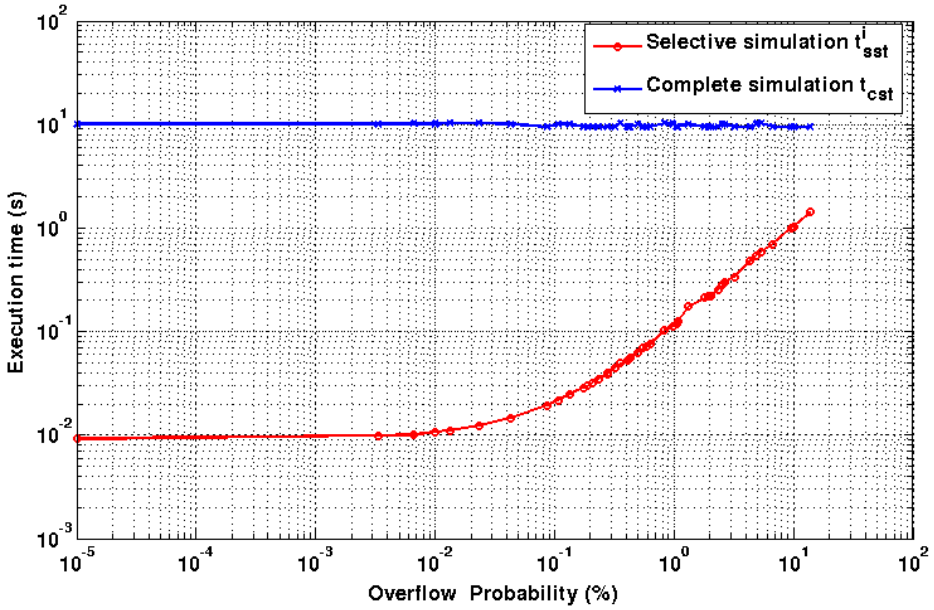


Figure 4.10 – Execution time according to overflow probability.

4.4.3.2 Execution time

The performance of the proposed approach is compared with the complete simulation approach in terms of execution time (duration). Figure 4.10 shows the durations t_{cst} and t^i_{sst}, defined in Equations 3.26 and 3.29, according to the overflow probability. The recorded durations are for different overflow probabilities, due to different IWL configurations. At the level of optimization, a tested IWL corresponds to one evaluation. This would give orders of magnitude in the case of optimization algorithm where high number of iterations are needed.

For the complete simulation technique, the execution time of any evaluation t_{cst} is equal to 10 s, which is independent of the overflow probability. However, the proposed approach needs significantly less time to evaluate the overflow effect. For low overflow probabilities, the gain in terms of acceleration is huge, where a gain of up to 1000 time is recorded. Naturally, when the overflow occurrence increases, the execution time increases. This increase is proportional to the number of overflows, as Figure 4.10 shows. In general, a small degradation of application quality is acceptable. Thus, quality evaluations are limited to the interval of low overflow probability and the gain in term of execution time will remain significant.

4.5 Conclusion

In this chapter, the implementation of the selective simulation framework in $C++$ is explained. The framework is tested on two application, GPS and the FFT part of the OFDM receiver. The experimental results show significant acceleration of the simulations, where the execution time is

reduced up to 1000 time with respect to complete simulation based approach. Moreover, the cost analysis on some SNRs per bit shows the possibility of reducing the implementation cost. In the next chapter, we present the proposed integer world length optimization algorithm that exploits the proposed selective simulation technique.

CHAPTER 5

Integer word-length optimization

5.1 Introduction

In the previous chapter, we explained the implementation of our proposed framework to accelerate the process of quality evaluation when overflows occur. Results obtained by testing the framework on two applications show the effectiveness of the proposed framework.

In this chapter, we use this framework to optimize the world-length of systems that uses fixed-point arithmetic. Especially, we focus on the optimization of the integer part word-length. The optimization problem is translated into the exploration of the trade-off between implementation cost and application quality. The cost and quality of fixed-point systems depend on the data word-lengths. The choice of fixed-point formats for each variable has to be done carefully to assure a good trade-off between the cost and the quality. After presenting the world-length optimization problem, we explain the existing algorithms to solve this optimization process. Then, we present our proposed algorithm and the conducted experiments to show the effectiveness of our algorithm.

5.2 Word-length Optimization problem

The floating-point to fixed-point conversion process has been mathematically formulated as an optimization problem [10]. Let C be the total cost function. Classical metrics for implementation cost are area, clock period, latency and power consumption. Let λ be the quality metric function. Considering an application with quality λ, the implementation cost C is reduced through minimizing the word-length, while maintaining the degradation of the application quality $\Delta\lambda$ lower than a maximal value $\Delta\lambda_{\max}$. The word-length optimization problem can be expressed as follows

$$\min\left(C(\mathbf{w_d})\right) \quad \text{such as} \quad \Delta\lambda\left(\mathbf{w_d}\right) < \Delta\lambda_{\max}, \tag{5.1}$$

where $\mathbf{w_d}$ is a N-length vector containing the word-length of each data, and N is the number of variables for which their word-length are optimized.

Fixed-point conversion aims at choosing a fractional part word-length w_f, which delivers a

sufficiently large computation accuracy, and an integer part word-length w that limits overflow occurrence. To determine the data word-length $w_d = w + w_f$, a trade-off between high numerical accuracy and overflow occurrence has to be investigated.

In classical fixed-point conversion techniques, the integer part word-length is computed such that no overflow occurs. Thus, determining the data word-length consists of two steps. First, the dynamic range of the different data is evaluated, and the number of bits for the integer part is computed.

Secondly, the number of bits for the fractional part is optimized such that the quantization noise is sufficiently low to maintain the application quality. Only the effect of quantization noise is considered and the optimization problem can be formulated as follows:

$$\min\left(C(\mathbf{w_d})\right) \quad \text{such as} \quad \Delta\lambda\left(\mathbf{w_f}\right) < \Delta\lambda_{\max}, \tag{5.2}$$

where $\mathbf{w_f}$ is a N-length vector containing the FWL of each variable. In this case, only the quality degradation due to quantization noise is studied.

Advanced fixed-point conversion techniques take into account the effect of overflows on the quality criteria. Both the IWL and FWL are minimized under quality constraints, and the optimization problem can be formulated as follows:

$$\min\left(C(\mathbf{w_d})\right) \quad \text{such as} \quad \Delta\lambda\left(\mathbf{w_f}, \mathbf{w}\right) < \Delta\lambda_{\max}, \tag{5.3}$$

where $\mathbf{w}$ is a N-length vector containing the IWL of each variable.

To obtain a reasonable complexity for the fixed-point conversion process, the determination of the IWL and the FWL are handled separately. Thus, the optimization process presented in equation 5.3 is split into two optimization processes: one for the FWL as presented in equation 5.2 and one for the IWL as presented in equation 5.4, it can be expressed as follows:

$$\min\left(C(\mathbf{w_d})\right) \quad \text{such as} \quad \Delta\lambda\left(\mathbf{w}\right) < \Delta\lambda'_{\max} \tag{5.4}$$

where $\Delta\lambda'_{\max}$ is a new application quality degradation constraint such that the quality degradation is budgeted between the two optimization processes.

For the two optimization processes presented in equations 5.4 and 5.2, the evaluation of the cost requires the knowledge of the global word-length $\mathbf{w_d}$. To overcome this problem, the fixed-point conversion process, presented in Figure 5.1, is applied. First, the dynamic range is evaluated guaranteeing no overflow, and a maximal value for each integer part word-length w^{IA} is obtained thanks to interval arithmetic technique. Let $\mathbf{w}^{IA}$ be a N-length vector representing the integer word-lengths of each data obtained with interval arithmetic technique. Secondly, the FWLs of all the variables are optimized, where w^{IA} is used to calculate the implementation cost. Third, the IWLs are optimized taking into account the overflow effect on the application quality, and the optimized FWLs obtained in the previous step are used to calculate the implementation cost.

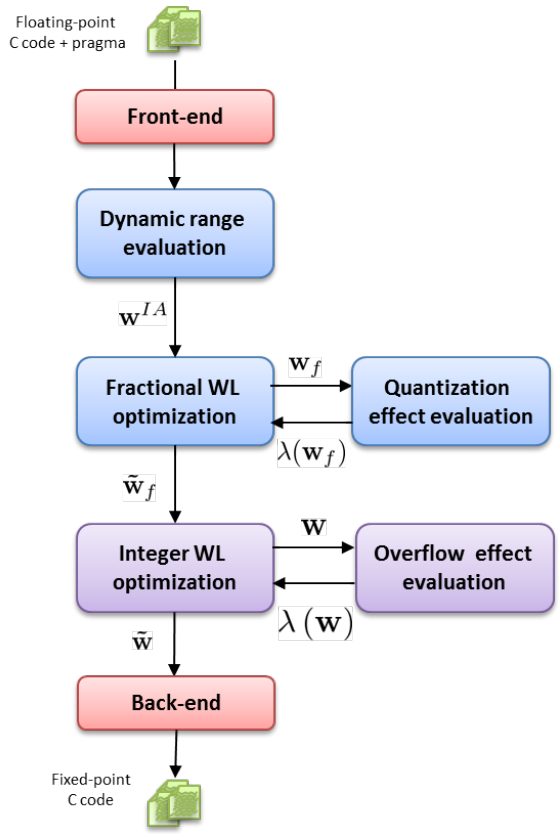


Figure 5.1 – Advanced fixed-point conversion process.

Optimization Variables A signal processing system consists of hundreds of fixed-point operations. For example, a $64 - point$ FFT signal processing algorithm has 960 fixed-point operations including addition and multiplication with constants. The integer and the fractional parts are chosen such that the fixed-point format accommodates the dynamic range of all signals and has enough precision. Moreover, it is possible to have flexible data while working on flexible architectures such as an FPGA, ASIC or modern-day DSP architectures with SIMD support. To solve the word-length optmization problem, the choice of word-lengths for each of the fixed-point operations has to be assigned optimally. Let M be the number of signals in the system. If for each of these signals, N different word-length configurations can be assigned, there can be N^M different word-length combinations. This is referred to as the multiple word-length assignment (MWL) paradigm for fixed-point refinement. The complexity of evaluating the cost-quality trade-off for each of these combinations is practically impossible.

Assigning the same word-length format for each signal reduces the complexity of the word-length optimization problem. This is the popular uniform word-length assignment (UWL) paradigm for fixed-point refinement. In this approach, all the computations are carried out with the same word-length. In other words, all the signals and operators in the system are assigned with the same fixed-point format. This approach for fixed-point design of signal processing systems was predominantly used for implementation on fixed-width data-path architectures. Then,

the integer and fractional parts of the fixed-point format is chosen such that the dynamic range of all signals are accommodated and there is enough precision for all signals. If the uniform word-length paradigm were to be used, the number of word-length combinations that are tested before reaching optimality is N , where N is the number of word-length configurations realizable on the given architecture. Although UWL paradigm significantly reduce the complexity of the problem , it may lead to overestimation of some world-lengths, and thus the implementation cost. A signal that requires a large world-length may affect the world-length(s) of other signal(s) if they can tolerate less accuracy.

5.3 State of the art

Fixed-point word-length optimization can be broadly approached in two directions depending on the optimization criteria. It could either be a cost minimization problem under an accuracy constraint or an accuracy maximization problem under a given cost constraint. The baseline principle for any product design is that the end product meets the design requirements. Improvement in its quality is generally secondary to its functional correctness. Keeping this point of view, the cost minimization problem rather than the accuracy maximization problem is considered in this thesis.

The word-length optimization problem is known to be combinatorial in nature, and arriving at an optimal solution to this problem is known to be NP-hard in complexity [91]. Given that every additional optimization variable causes an exponential increase in the combinatorial search space, the scale of the word-length optimization problem can easily grow beyond manageable limits. Such problems are usually solved by using heuristic guidelines iteratively. However, every iteration of the optimization process can be very time consuming as it requires the evaluation or estimation of the numerical accuracy and the total cost for every tested world-length.

Consider the optimization problem presented in equation 5.1, where w_d is a discrete vector that represents the word-length of each variable. In this problem, the relation between the quality, $\lambda(w_d)$, and the cost, $C(w_d)$, is not necessarily monotone. In this regards, the search for optimal data width is a problem of discrete optimization under constraint and multivariate. This class is called combinatorial optimization problems. Although combinatorial optimization problems are easy to identify and resolve by listing all the solutions, the time required to solve the problem is often prohibitive and optimization algorithms in reasonable time are needed. In the rest of this section, different classes of available algorithms to solve this optimization problem are summarized. These techniques have been widely used to solve the fractional word-length optimization problem as depicted in Equation 5.2.

5.3.1 Deterministic algorithm

Exhaustive search The method proposed in [70] performs an exhaustive search in a subspace of the search space. The algorithm begins with the minimal word-length combination (MWC), which is obtained by decreasing its word-length until the accuracy constraint is fulfilled considering all the other variables with their maximal word-length (32 bits as example), as depicted in Algorithm

1. To simplify the notation, the vector of the total word-length $\mathbf{w_d}$ is named $\mathbf{w}$

Algorithm 1 Determination of minimal word-length combination

$w^{\max} = 32$
$\mathbf{w}^{\max} \leftarrow [w^{\max}, \dots w^{\max} \dots w^{\max}]$
for all $1 \leq k \leq N$ **do**
  $\mathbf{w} \leftarrow \mathbf{w}^{\max}$
  while $\lambda(\mathbf{w}) \leq \lambda_{min}$ **do**
    $\mathbf{w}_k = \mathbf{w}_k - 1$
  end while
  $\mathbf{w}^{\min} = \mathbf{w}_k + 1$
end for

With the MWC, the accuracy constraint may not be fulfilled. Thus, a bit is temporarily added to a variable and the accuracy constraints is tested. If this constraint is not fulfilled, then the total number of bits added to the MWC and distributed to various variables is incremented. The procedure is repeated as long as the accuracy constraint is not satisfied.

This method does not take into account the actual implementation cost of selected operations to increase the word-length. Thus, the word-length of some expensive operations may be increased at the expense of less expensive operations.

Greedy search A greedy algorithm is an iterative algorithm that makes locally optimal choice at each iteration (stage) with the hope of finding a global optimum. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally-optimal solutions that approximate a global optimal solution in a reasonable time.

min + 1 bit is a greedy algorithms that uses sequential search [92]. The main idea is to add 1 bit to the variable that leads to the best performance at the current stage. *min+b bits* is a generalized form of *min + 1 bit* that is used to avoid local maxima and converge more rapidly, where b bits are distributed in total at each stage. Other *min + 1 bit* variants are the algorithms *max - 1 bit* and *max - b bit* [93]. These algorithms start with the combination of maximum word-length $\mathbf{w}^{\max}$. Algorithms *min + 1 bit* and *max - 1 bit* ensure that solutions are 1-optimal, while *min+b bits* and *max - b bits* variants do not guarantee anything. The algorithm *max - 1 bit* is presented in Algorithm 2.

If a solution at a distance d is found, the number of evaluations is Nd. In the worst case, the number of steps is equal to:

$$\sum_{k=1}^{N} (\mathbf{w}^{\max} - \mathbf{w}^{\min}) \tag{5.5}$$

To improve the search for the best direction in every step of the greedy algorithm, it is possible to compute a metric q_k for each direction k and to select the one leading the best direction. This metric, computed with the function f_{dir} in Algorithm 2, can be the ratio of the gain in quality

and the increased in cost. Moreover, greedy algorithms can be used in cooperation with other algorithms [94] to enhance the solution.

A modified greedy algorithm is proposed in [95]. The algorithm limits the number of explored solutions at each stage to K. This means that the procedure at a specific stage ends when the number of tested solutions reaches the limit K. Moreover, the algorithm memorizes all the solutions leading to cross the quality constraint, which ensures that all tested possibilities in different iterations are taken into account, and not only those that are on the way. The result is better, or at worst equal to that obtained with a classical greedy algorithm.

Algorithm 2 $Max-1$ bit

$\mathbf{w} \leftarrow \mathbf{w}^{\max}$
while $\lambda(\mathbf{w}) \leq \lambda_{min}$ **do**
  for all $1 < k \leq N$ **do**
    $q_k \leftarrow f_{dir}(w_k)$
  end for
  $i = \text{argmax} \quad q_k$
  $\mathbf{w_i} \leftarrow \mathbf{w_i} - 1$
end while

Heuristic process A heuristic process was proposed by W. Sung in [70]. The goal is to limit the number of iterations in order to reduce the number of quality and cost evaluations. Reducing the number of iterations is crucial when simulation-based techniques are used to evaluate the accuracy. In the proposed algorithm, all word-lengths (of variables to be optimized) are simultaneously increased starting with the MWC until the quality constraint is satisfied. This step normally takes a few simulations. The next step is equivalent to the *max-1 bit* algorithm, where one word-length is reduced at each iteration starting by the most expensive operation. This step is repeated as long as the constraint is satisfied and it requires N quality evaluations. The procedure assumes that the solution is near the MWC, but this assumption is not always valid.

Local search Local search algorithms are used to find a local minimum from a starting point. The algorithm begins from a starting solution and iteratively moves to a neighboring solution. In a word-length optimization problem, two configurations are considered neighbors of distance d if the difference between their word-lengths is at most equal to d. The optimal solution is called $d-optimal$.

A local search can stop after a fixed number of iterations or when the best solution found by the algorithm has not been improved after a given number of iterations. In the second case, the solution may not be the best in the search space. Thus, local search algorithms are then sub-optimal algorithms.

Tabu search Tabu search (*TS*), first introduced by Glover [96, 97], is a heuristic procedure to find good solutions to combinatorial optimization problems. This technique is usually combined with other methods. In many cases, the coupling of the greedy algorithm with tabu search allows a better local search and avoids unnecessary moving. A tabu list T is defined to contain all the solutions from the last t iterations. Let x be the current solution and $N(x)$ its neighborhood. $N(x)$ contains all the possible moving from x. At each step, the procedure moves to the solution in $N(x)$ leading to the best improvement of the objective function. Then, the tabu list and the best solution are updated. The solution is removed from the tabu list T. The tabu list avoids returning to the local optimum from which the procedure has recently escaped.

In [98] a hybrid heuristic procedure, which selectively increments or decrements the word-length to converge to an optimized solution, is explored. The ratio of the gradient of the cost and the gradient of the accuracy is used to choose the best solution for the next iteration. The direction is reversed to a decrement mode when the solution leads to an accuracy higher than the constraint. The variables that doesn't contribute to the right direction are moved to the tabu list.

Branch and Bound The branch and bound algorithm (BaB) is an implicit enumeration method. It is based on the knowledge of the problem properties, which allows to list the potentially good solutions to be tested. This algorithm is a general method that can be applied individually or in combination with other methods.

In [99], the BaB algorithm is used directly, but this is only valid for small and medium sized problems, due to its exponential complexity. In [100] a methodology is proposed to reduce the search space taking into account the capability of the targeted architecture. The minimum cost and the maximal quality of each branch is evaluated. The branches having a minimal cost higher than the cost of the current solution or a maximum quality not satisfying the quality constraint are ignored.

Integer Linear Programming In this type of techniques, the optimization problem is relaxed into an integer linear programming (*ILP*) or mixed integer linear programming (*MILP*) by formulating the cost and constraint(s) functions in a linear form. Then, a heuristic search is used to solve the problem as the ILP (and MILP) is NP-hard. The most widely used technique to solve ILP is BaB. This solution is proposed in [101, 102]. It leads to a long optimization time when the number of variables is high. Thus, this technique is rather used to compare the efficiency of different optimization algorithms on simple applications.

5.3.2 Random algorithms

Random algorithms have also been used for solving the word-length optimization problem. At each iteration of the optimization algorithm, the solutions are generated randomly and the best are kept for the next iteration.

Simulated annealing Simulated annealing (SA) technique is widely used in word-length optimization. In [103], the word-length optimization problem is formulated as a mixed integer linear problem and SA is used to find the solution. The error power and the implementation cost are evaluated every time the values of the variables change (movement). The various movements can be thought of as generating different states in a Markov-chain. At each step, the SA heuristic considers some neighbouring state s' of the current state s, and probabilistically decides between moving the system to state s' or staying in state s. These probabilities ultimately lead the system to move to states of lower energy. Typically this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted. A probability is assigned to the solution generated by the movement depending on the quality of the solution obtained, which is a sort of recommendation on whether or not to jump to the new solution. In [104], an adaptive simulated annealing procedure is proposed along with the use of minimum word-length. However, this procedure can take long period of time in very large designs.

Genetic algorithm Genetic Algorithms (*GA*s) are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetics. As such, they represent an intelligent exploitation of a random search used to solve optimization problems. GAs exploit historical information to direct the search into the region of better performance within the search space. The population of individuals within the search space represents the possible solutions to a given problem. Each solution is coded as a finite length vector of components, or variables. The solutions are similar to chromosomes and the variables are analogous to genes. Thus a solution is composed of several variables. A fitness score is assigned to each solution representing the abilities of an individual to 'compete'. The individual with the optimal (or generally near optimal) fitness score is sought. The GA aims to use selective 'breeding' of the solutions to produce 'offspring' better than the parents by combining information from the chromosomes. Once the solution is obtained, the survivability of the solution depends on its performance. In [105], genetic algorithms have been adapted for word-length determination of the Least Mean Square (LMS) filter. In [98], a hybrid technique using a gradient based heuristic approach with the GA approach is proposed. This technique leads to faster convergence.

5.4 IWL optimization algorithm

Numerous algorithms have been proposed to solve the fractional word-length optimization problem as depicted in Equation 5.2. However, optimizing the integer world-length (IWL) is rarely tackled. Moreover, simulation-based approaches are usually time consuming. Thus, a time efficient technique is needed to evaluate the application quality degradation due to overflow.

In this section, an algorithm is proposed to solve the integer word-length optimization problem as depicted in Equation 5.4. This algorithm exploits the selective simulation technique, proposed in Chapter 4, to accelerates the evaluation of the application quality degradation due to overflow. The solution $\mathbf{w}^{\mathbf{IA}}$ obtained from interval arithmetic or affine arithmetic is used as a starting point, and an iterative procedure is applied to converge to an optimized solution.

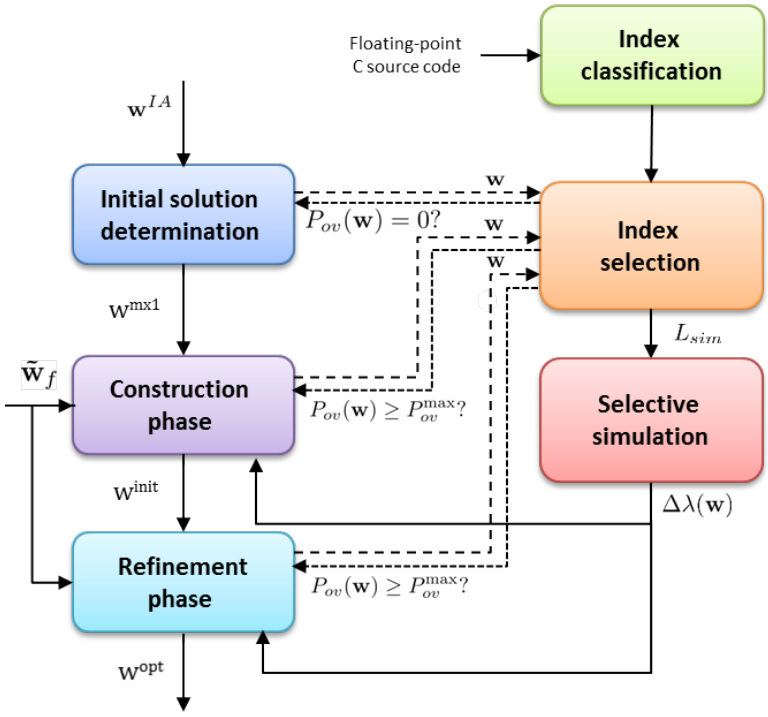


Figure 5.2 – Proposed IWL optimization algorithm combined with selective simulation for overflow effect analysis.

The IWL optimization is carried-out with a greedy based algorithm that minimizes the implementation cost as long as the performance degradation constraint is satisfied. Figure 5.2 presents the proposed optimization algorithm combined with the proposed selective simulation based approach for overflow effect evaluation. The proposed algorithm consists of three phases: initial phase, construction phase and refinement phase. The first phase leads to an initial solution for which the performance degradation is still null. In the second phase, which corresponds to the construction phase, a steepest descent algorithm (*max-1 bit*) is used for finding a sub-optimal solution w^{mx1}. Then a local search using Tabu search algorithm is applied to refine the solution in the third phase. At each iteration, the best direction is selected and the corresponding variable is modified to converge on an optimized solution. The phases of the IWL optimization algorithm are detailed in the following.

5.4.1 Initial phase

The aim of the initial phase is to find a starting solution for the construction phase. This phase starts with the solution obtained with interval arithmetic (IA) $\mathbf{w}^{IA}$. As IA overestimates the IWL, one or several IWLs may exist for a variable k that are lower than w_k^{IA} and with null overflow probability P_{ov}. These IWL configurations are detected using the *Index Selection (IS)* step, where the number of overflow occurrence is null (the list L_{sim} contains no element). The initial phase of the algorithm is presented in Algorithm 3, where the IWL $\mathbf{w}_k$ of each variable is decreased until

obtaining the minimum IWL with null P_{ov}, *i.e.* quality degradation is null. The solution obtained in this phase is $\mathbf{w}^{init}$.

Algorithm 3 Initial phase

$\mathbf{w} \leftarrow \mathbf{w}^{IA}$
for all $1 \leq k \leq N$ **do**
  while $P_{ov}(\mathbf{w}) = 0$ **do**
    $\mathbf{w}_k = \mathbf{w}_k - 1$
  end while
  $\mathbf{w}_k^{init} = \mathbf{w}_k + 1$
end for

5.4.2 Construction and refinement phases

The construction phase is based on a steepest descent greedy algorithm (max-1 bit), *i.e.* the IWL of each variable is reduced while satisfying the performance criterion. Starting from $\mathbf{w}^{init}$, this phase allows obtaining a sub-optimal solution. Then, a refinement around this solution is applied to improve the quality of the solution in the refinement phase. The latter is carried-out with a Tabu-search algorithm. The combination of greedy algorithm and Tabu search may achieve better local search and avoid unnecessary movements [98]. The proposed algorithm for construction and refinement phases is presented in Algorithm 4.

Criterion for direction selection At each iteration of the algorithm, the IWL of specific variable is modified to move toward the final solution. To select the best direction, a criterion has to be defined. We consider a criterion that computes the gradient of the application quality as follows:

$$f_{\nabla}^{\lambda}\left(\mathbf{w}^{k\pm}, \mathbf{w}^{k}\right) = \frac{\lambda\left(\mathbf{w}^{k\pm}\right) - \lambda\left(\mathbf{w}^{k}\right)}{\|\mathbf{w}^{k\pm} - \mathbf{w}^{k}\|} \tag{5.6}$$

where $\mathbf{w}^{k} = [w_0, ..., w_k, ...w_{N-1}]$ and $\mathbf{w}^{k\pm} = [w_1, ..., w_k + d, ...w_{N-1}]$ are the word-length vectors before and after a possible modification at position k respectively. The term d represents the direction and is equal to 1 for steepest ascent algorithm (*min+1*) and -1 for steepest descent algorithm (*max-1*).

To improve the decision of the best direction selection, the cost due to the IWL modification is taken into account. This will give a good trade-off between the implementation cost C and the application quality. The new criterion selects the direction which minimizes the cost increase and maximizes the application quality increase as follows:

$$f_{\nabla}^{\lambda/C}\left(\mathbf{w}^{k\pm}, \mathbf{w}^{k}\right) = \frac{\lambda\left(\mathbf{w}^{k\pm}\right) - \lambda\left(\mathbf{w}^{k}\right)}{\mathcal{C}\left(\mathbf{w}^{k\pm}\right) - \mathcal{C}\left(\mathbf{w}^{k}\right)} \tag{5.7}$$

Algorithm 4 Tabu search in IWL optimization

1: $T \leftarrow \emptyset$ {Empty list of tabu variables}
2: $\mathbf{w}^{opt} \leftarrow \emptyset$
3: $d \leftarrow -1$ {set the direction for steepest descent}
4: **while** $|T| < N$ **do**
5: **for all** $1 \leq k \notin T \leq N$ **do** {calculate criterion}
6: $\mathbf{w}^{k\pm} = \mathbf{w}^k + d$
7: **if** $P_{ov}(\mathbf{w}^{\mathbf{k}\pm}) \geq P_{ov}^{\max} \vee w_k^{\pm} \notin \mathcal{W}_k$ **then**
8: $T \leftarrow T \cup \{k\}$
9: **else**
10: $\nabla_k \leftarrow f_{\nabla}\left(\mathbf{w}^{k\pm}, \mathbf{w}^{\mathbf{k}}\right)$
11: **end if**
12: **end for**
13: **if** $|T| < N$ **then**
14: **if** $\mathrm{d} > 0$ **then**
15: $j \leftarrow \mathrm{argmax}\nabla_k$ {Steepest ascent alg. }
16: $\mathbf{w}_j \leftarrow \mathbf{w}_j + 1$
17: **if** $\Delta\lambda(\mathbf{w}) \leq \Delta\lambda_{\max}$ **then**
18: $d \leftarrow -1$
19: $T \leftarrow T \cup \{j\}$
20: **end if**
21: **else**
22: $j \leftarrow \mathrm{argmin}\nabla_k$ {Steepest descent alg. }
23: $\mathbf{w}_j \leftarrow \mathbf{w}_j - 1$
24: **if** $\Delta\lambda(\mathbf{w}) > \Delta\lambda_{\max}$ **then**
25: $d \leftarrow 1$
26: **end if**
27: **end if**
28: **end if**
29: **end while**
30: $\mathbf{w}^{opt} \leftarrow \mathbf{w}$
31: **return** $\mathbf{w}^{opt}$

Algorithm description At each iteration, the procedure moves to one of the neighbourhood of the current solution $\mathbf{w}$ according to the value of d. The Tabu list T is the set of variables no longer used. This list is updated at each iteration to avoid useless or infinite loops. The next position of $\mathbf{w}$ for each variable k not belonging to T is calculated in line 6 of Algorithm 4. Let $\mathcal{W}_k$, be the set of valid values for the variable k. For each variable k, the set of valid values is bounded by w_k^{init}

as maximum and 1 as minimum. If the next position $w_k^{\pm}$ of the variable k does not belong to $\mathcal{W}_k$ or leads to an overflow probability greater than the maximum value $P_{ov}^{\max}$, the variable k is added to the Tabu list T as shown in line 8. If the movement is valid, the criterion for direction search associated with this variable is evaluated. The index selection step allows the evaluation of the overflow probability as shown in Figure 5.2. This would result in avoiding unneeded simulations when overflow occurrence is too high, which leads to a gain of time.

Line 13 verifies whether the set T is complete or not. If the set is not complete, the variable with the best criterion is selected (lines 14 to 29). Then, its IWL is increased or decreased by one bit, depending on the selected direction. When $\Delta\lambda(\mathbf{w})$ exceeds the constraint $\Delta\lambda_{\max}$, the direction d is reversed. The iterative process stops when the set T is complete.

5.5 Experiments and results

In this section, the efficiency of our approach for IWL optimization is evaluated through several experiments. First, the cost and quality trade-off is analyzed to show that cost reduction can be obtained with IWL optimization. Secondly, the optimization time is analyzed and compared with the classical approach. The proposed approach is tested on the GPS application presented in section 4.4.1 and on the OFDM transmission chain presented in section 4.4.2.

5.5.1 Cost reduction

5.5.1.1 GPS application

We study the trade-off between the system cost and the degradation of the application quality criteria. For the GPS application, to simplify, the cost is computed as the sum of the number of bits of the integer part assigned for each studied variable, and then normalized to the cost obtained by IA.

Figure 5.3 shows the variation of the normalized cost in the GPS application with respect to the normalized distance error δ_d. The curve starts with a significant reduction of the cost (25%) for a slight degradation of the application quality. Then, more degradation of the application quality is required to achieve higher cost reduction. Afterwards, the cost reduction comes with high price in terms of application quality degradation.

5.5.1.2 OFDM receiver

Experiments have been also conducted for the OFDM application presented in section 4.4.2. The IWL optimization algorithm has been applied on the FFT part, of the OFDM receiver. The energy consumption metric is used for calculating the implementation cost C. A library of characterized operators for FPGA target [106] is used to compute the cost according to the word-length $\mathbf{w_d}$. The FWL is fixed to 8 bits for all the variables to calculate $\mathbf{w_d}$. The performance degradation

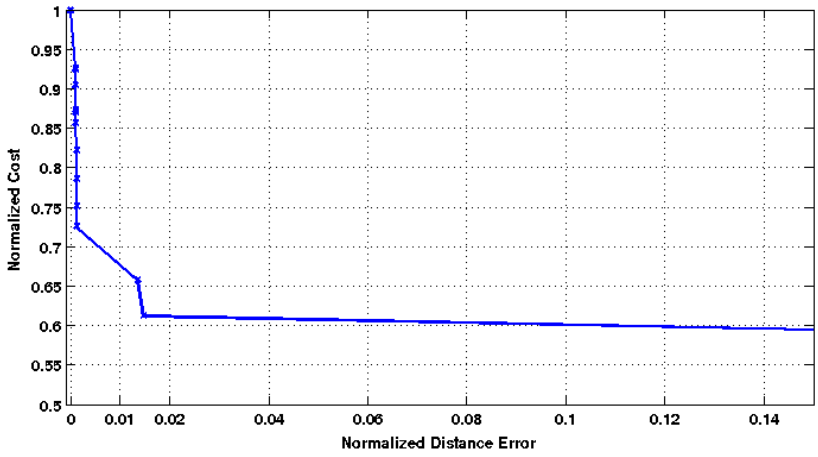


Figure 5.3 – Normalized cost according to normalized distance error (δ_d).

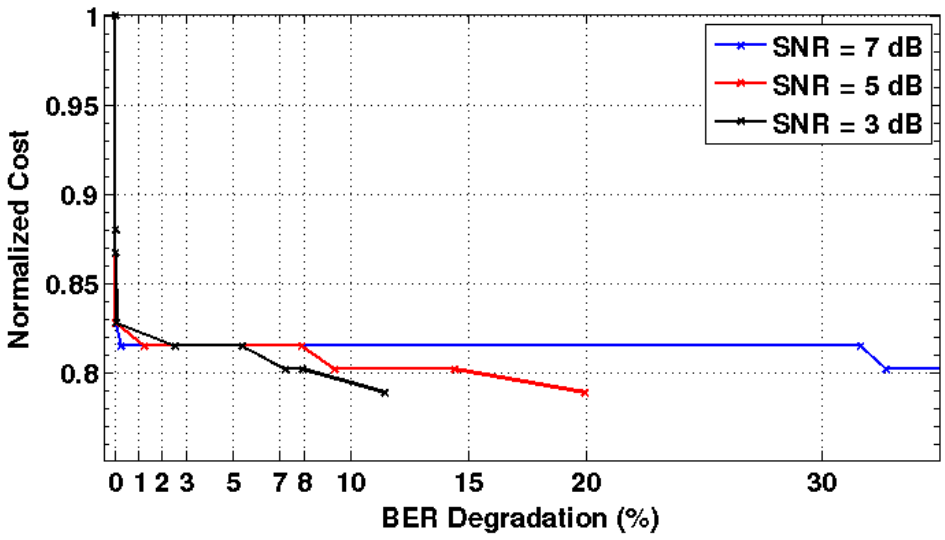


Figure 5.4 – Pareto-curves of the normalized cost C_{opt} according maximum BER degradation $\Delta BER_{\max}(\%)$ for different SNRs.

is evaluated through the Bit Error Rate (BER) degradation $\triangle_{BER}$.

The IWL optimization is carried-out for different BER degradation constraints corresponding to the term $\Delta\lambda'_{\max}$ in equation 5.4. The optimized cost C_{opt}, obtained by our IWL optimization algorithm, is normalized to the cost obtained with interval arithmetic. Figure 5.4 represents the curves of the normalized cost as function of the maximum BER degradation. The curves evolve by levels leading to a step curve due to the integer values of the IWL.

The obtained curves are Pareto curves decomposed into three parts. The first part of each curve can be assimilated as a line characterized by a very high slope. The slope of the curve depends on the probability density function (*PDF*) of the application input data. In this example, the input data follows a platykurtic distribution [107] and the short tails of the PDF determines the very high slope. This part shows that the implementation cost can be significantly reduced compared to the starting solution $\mathbf{w}^{IA}$, with a very low BER degradation. The pessimistic solution obtained with interval arithmetic explains this phenomena. The second part of each curve corresponds to the "bend". This zone is the interest of the designer, since it represents a good trade-off between the cost and the quality. For the three SNR chosen as example, the implementation cost is reduced between 16% and 18% with a low BER degradation. In the third part, the implementation cost can be reduced but comes at a price of a high BER degradation.

5.5.2 Optimization time enhancement

Comparisons between three algorithms have been carried-out to verify the time efficiency of the proposed algorithm. *Opt+Ssim* corresponds to the proposed optimization algorithm described in Section 5.4, where the selective simulation technique is used to evaluate the overflow effects as detailed in the previous chapter. *Mx1+Csim* is a classical steepest descent algorithm (max-1) that uses complete simulation technique to evaluate the overflow effects. It represents the reference case, where all the samples are simulated. *Opt+Csim* combines our IWL optimization algorithm and complete simulations. For the three algorithms, the starting solution is $\mathbf{w}^{IA}$ obtained by interval arithmetic.

Figure 5.5 shows the evolution of the optimization time of the three algorithms for different SNR per bit with respect to the BER degradation constraint $\Delta BER_{\max}$. Results show that *Mx1+Csim*, the reference algorithm, is the most time consuming. Using *Opt+Csim* reduces the optimization time up to 2.8 times. However, *Opt+Ssim* accelerates significantly the optimization time, where an acceleration factor between 72 and 617 is reported with respect to *Mx1+Csim*. Moreover, *Opt+Ssim* reduces the optimization time between 43 and 176 times with respect to *Opt+Csim*. These results emphasize the efficiency of the proposed optimization algorithm, especially when it is combined with selective simulation technique.

For different SNR per bit, the number of iterations in the optimization algorithms increases when the BER degradation constraint is relaxed *i.e.* high $\Delta BER_{\max}$. This results in an increase of the optimization times of the three algorithms. Moreover, the increase of $\Delta BER_{\max}$ results in higher overflow occurrence. This increases the number of indices to be simulated (size of L_{sim}) in the case of *Opt+Ssim* and thus increases the simulation time of each iteration., and relatively higher optimization times are observed.

For the different possibilities of SNR per bit and $\Delta BER_{\max}$, our algorithm provides a solution leading to the same or a better cost in comparison with *Mx1+Csim* while accelerating the optimization time.

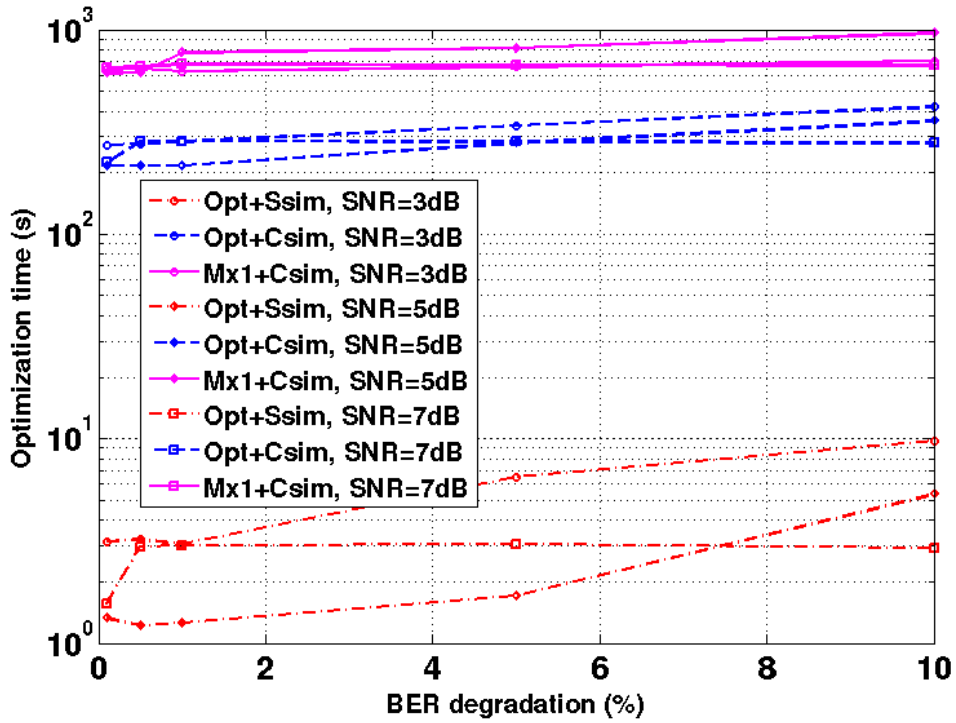


Figure 5.5 – Optimization time evolutions (s) according to the maximum BER degradation $\Delta BER_{\max}(\%)$ for different SNRs.

5.6 Conclusion

In this chapter, we have presented the world-length optimization problem. In contrast to classical fixed-point conversion, advanced conversion techniques take into account the effect of overflows on the quality criteria. The fixed point conversion is formulated as an optimization problem, which appears to be a combinatorial problem with high complexity. Thus, heuristic algorithms are used to solve this problem to meet time constraints and minimize the cost.

After presenting the existing approaches for solving the world length optimization problem, we present our proposed IWL optimization algorithm that uses the selective simulation technique to evaluate the overflow effects on application quality. This algorithm does not only reduce the cost, but also allows overcoming the long execution time of classical simulation based algorithms. Through experiments applied on the FFT part of an OFDM chain, the proposed algorithm results in a significant reduction of cost with acceptable degradation of quality criteria. At the same time, results show important enhancement in the optimization time, where the acceleration factor reaches up to 617 with respect to max-1 bit.

CHAPTER 6

Quality Evaluation in Presence of Decision Error

6.1 Introduction

In the previous chapters, the work was focused on the study of overflow effects and the use of selective-simulation technique to reduce the execution time of the quality evaluation in presence of overflows. This chapter considers the use of selective-simulation technique in the presence of decision errors. Firstly, the model of single noise source is presented and the effect of un-smooth operation is explained. Secondly, the adaptation of selective-simulation technique in the presence of decision error is detailed. Finally, we present the experiments and results that show the effectiveness of the proposed technique.

6.2 Techniques for Quality Evaluation

The reduction of the number of bits in a fixed-point system leads to a quantization error. When the size of the system grows, the number of operations and noise sources increase, which results in a longer time to evaluate the quantization noise power of the system output. Moreover, the increase in the number of noise sources leads to an increase in the number of variables to be optimized resulting in a higher complexity for word-length optimization process, and thus the evaluation of fixed-point precision accuracy has to be carried out more often.

6.2.1 Single noise source model

The pseudo-quantization noise (PQN) model, also called quantization noise model of Widrow [22], defines a statistically equivalent random process to the uniform quantization of a signal x. In this model, the effect of uniform quantization of a signal x can be represented by an additive white noise b_x. The addition of b_x to the signal x results in $\hat{x}$, which is statistically equivalent to the signal obtained after the uniform quantization of the signal x. Thus, the errors generated by the fixed-point operation can be considered as a random process. The Single Noise Source (*SNS*), described in [21], model extends the PQN model by modelling the total quantization noise in a

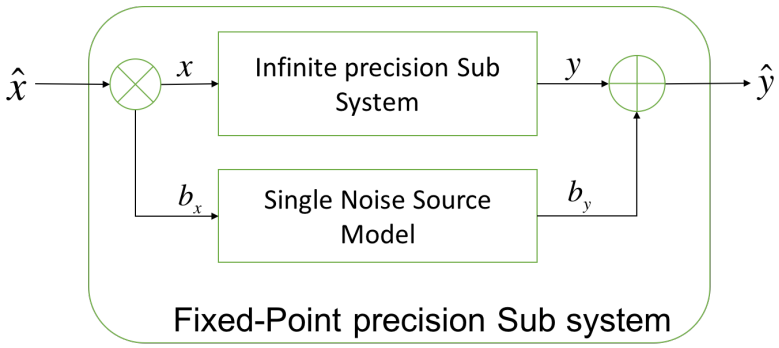


Figure 6.1 – Abstraction of the Single Noise Source model.

sub-system. The SNS model determines the statistics of the quantization noise at the output of the sub-system due to the use of fixed-point operations. The total quantization noise at the output of a fixed-point sub-system is the result of the noise generated by the fixed-point operations in the sub-system and the noise generated in previous sub-systems and propagated through the sub-system under consideration.

Consider a sub-system B, as shown in Figure 6.1, with input $\hat{x} = x + b_x$ and output $\hat{y} = y + b_y$ implemented using fixed-point operations. The input x and the output y of the infinite precision sub-system are perturbed by the quantization noise b_x and b_y due to fixed-point implementation. The PQN model requires that the signal and the quantization noise are uncorrelated. Therefore, the input and output quantization noise (b_x and b_y) are uncorrelated respectively with the input and the output signal (x and y). In the SNS block, the noise b_x is propagated through the sub-system giving the propagated noise b_x^t. The noise b_g is the noise generated within the sub-system due to fixed-point operations. The noise b_y is the sum of b_x^t and b_g.

6.2.2 Identifying un-smooth operations

Concept of un-smooth operation The concept of smoothness of an operation is the basis of the perturbation theory. As defined in [108], a deterministic function is said to be smooth if this function is continuous and differentiable to any desired degree over an open set in which the function inputs belong to. An operation O is called smooth over its inputs, or briefly smooth, if its instantaneous transfer function is smooth. In [108], it has been shown that differentiability of order 3 is always sufficient in order to model the quantization effects. The smooth or un-smooth character of an operation does not make sense when its inputs correspond to logical signals. Basic operations such as adders or multipliers are smooth. Moreover, according to [108], an operation is un-smooth on a given domain if its characteristic function is not of class C^1 on this domain. According to these definitions, the decision operations with arithmetic inputs and logical outputs are un-smooth operations, but they can be treated as smooth on a particular definition domain where the inputs belong to the same decision region. The un-smooth regions contain the inputs that produce different decisions by adding an infinitely small perturbation.

In terms of quantization, we may consider an operation as un-smooth if the deviation of its observed quantization error statistics is sufficiently important compared to the error statistics determined by the PQN model. The applicability of the PQN model to study the quantization process can be deduced by comparing the dynamic range D of the input signal and the quantization step-size q. If q is much smaller than D, it is possible to model the errors due to quantization using the PQN model. When the quantization size is large and comparable to the dynamic range of the signal, the operation becomes un-smooth. In this case, the properties of the quantization error including the noise shape, its additivity properties and its independence with input signal statistics are no more valid.

Identifying Un-smooth operation In a practical scenarios, a fixed-point arithmetic based system consists of a number of quantization operations. Analytical estimation obtained using the PQN model may be incorrect when applied to the quantization of an un-smooth operation. Therefore, it is important to classify the operations as smooth or un-smooth based on the characteristics of the signal to be quantized and the quantization step. In [109], a technique based on the calculation of the characteristic function to check the applicability of the PQN model in the case of any operation of uniform quantization is proposed. This technique identifies the bounds of the quantization step so that the PQN model can be applied to model the error behavior. The implementation using fixed-point arithmetic leads to the quantization of the operation inputs. The effect of the quantization of the input is covered by the quantizer at the output of the operation from which the signal comes. Therefore, a quantizer is associated with each signal. The effect of quantization of the input has an impact on the behavior of the error at the output of the operation under consideration. However, repeated quantization does not fundamentally change the dynamics. Thus, the technique works with the statistics of double precision data to evaluate the smoothness of each quantisation operation.

6.2.3 Quantization noise effect evaluation

Two approaches are used to evaluate the effect of the quantization noise at the output of a fixed-point system.The first one is simulation based approach which consists in evaluating the metric by simulating the system in fixed point precision and comparing the output with the reference output corresponding to the floating-point simulation. However, this approach requires a large simulation time to explore the fixed-point design space. For this reason, a second approach is proposed to deal with this problem. This approach consists in evaluating the precision metric using analytical methods by propagating a noise model within the system.

6.2.3.0.1 Analytical techniques for smooth operations Analytical techniques usually uses the combination of two models. On one hand, the PQN model to evaluate the quantization nosie stastistics at the output of fixed-point quantizer. On the other hand, the noise propagation model based on the perturbation theory to determine the influence of the fixed-point representation of the input on the computations.

PQN model works properly as long as the quantization step remains small compared with the dynamics of the signal as in the case of fixed-point arithmetic operations. In the case of un-smooth operations, quantization errors are very large and often comparable to the dynamics of the quantized signal. In this case, PQN model are no more valid and therefore the quantized signal will be far from the true value. The Min() and Max() operations are good examples for such kind of un-smooth operations. These operations are used to determine the minimum and maximum values respectively from data inputs. The statistical prediction of the outcome of minimum value and maximum value operation in spite of the knowledge of signal statistics is difficult when there are more than two inputs, and the analytical modelling of its fixed-point behavior becomes challenging.

Moreover, the analytical approaches based on the perturbation theory use Taylor series developments to evaluate the application quality. Thus, they require that the function associated with the operation to be differentiable on its domain. When the operation associated function is continuous but not differentiable, such as the absolute value function, the magnitude of the error output of the operation due to the input noise is of the same order of magnitude as that of the input noise. Thus, it is possible to obtain a propagation model for the noise, and technique based on perturbation theory are applicable. When the operation associated function is not continuous, the magnitude of the output error of the operation related to the input noise is not of the same order of magnitude as that of the input noise. Thus, the technique based on perturbation theory can no longer be used.

6.2.3.0.2 Simulation-based techniques for un-smooth operations Consider the conditional operation O with input c and output y and with N_{dec} decision boundaries. The output y has a value equal to y_k if the input $c \in E_k$, where $k \in [1, N_{dec}]$. Let c and y be the infinite precision values of respectively the input and the output. e_c and e_y are the input and output quantization errors respectively. These errors are the difference between the finite and infinite values. The output error depends on the decision of the operation in both finite and infinite precisions as follows:

$$e_y = \begin{cases} \hat{y}_k - y_k & if \quad \hat{c} \in E_k, \quad c \in E_k \\ \hat{y}_l - y_k & if \quad \hat{c} \in E_l, \quad c \in E_k \ \forall k \neq l \end{cases} \tag{6.1}$$

In the first case in Equation 6.1, the operation decisions in finite and infinite precisions are the same. In the second case, the operation decisions in finite and infinite precisions are different and a decision error occurs. The amplitude of the error e_y is high and its propagation in the rest of the system can not be treated with the analytical approaches based on perturbation theory.

In [5] an analytical model for the decision operations is proposed. The model determines the probability density of the error e_y. The expression of the PDF depends on the input c distribution of the decision operation and the distribution of the input quantization noise e_c. The expression of the PDF was developed in the case where the input c and the noise e_c are Gaussian. This approach evaluates finely the performance of an application containing a decision operation at its output. However, when the decision operation is located in the middle of the application, the propagation of the error e_y within the application and the evaluation of the correlation between the errors become a complex task. Under such circumstances, using fixed-point simulation to evaluate the

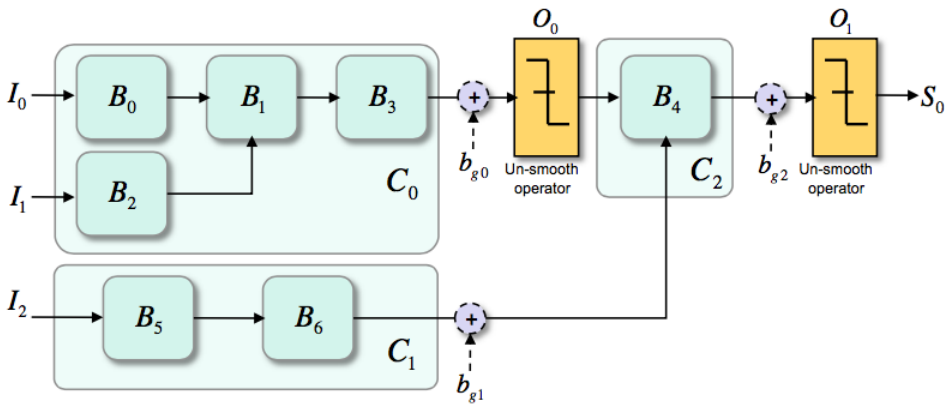


Figure 6.2 – Modelling of a system B composed of smooth and un-smooth oprations [6].

performance of fixed-point systems is inevitable.

During fixed-point simulation, all operations are simulated using a fixed-point library. When the use of fixed-point does not lead to a decision error at the output of the un-smooth operations there is no need to carry-out a complete simulation of the entire system to evaluate the quality of the system. It is important to determine whether an operation does encounter an error at its output for each input samples. This will lead to selectively simulate the system, which will result in gain of time.

6.2.4 Hybrid technique

The analytical techniques can greatly reduce the evaluation time, but are limited in terms of supported systems. Performance evaluation techniques based on simulation can be applied on all types of systems but lead to higher evaluation time. If the system under consideration consists of only smooth operations, it is possible to avoid performing fixed-point simulation by using analytical techniques. However, the presence of even one un-smooth operation or sub-system whose fixed-point behavior cannot be captured makes the use of fixed-point simulation for the entire system inevitable. In such situations, an un-smooth operation becomes an obstacle for taking the benefit of analytical models. Hybrid techniques using both analytical and simulation approaches are a good solution to deal with this probem.

In [21], a mixed approach combining analytical and simulation approaches is proposed. The goal is to use techniques based on simulation only when un-smooth errors occur and use the analytical results in other cases. This approach uses the system modeling illustrated in the example presented in Figure 6.2 and explained in the following.

System modeling Consider the system B presented in Figure 6.2. This system is composed of N_o decision operations O_i and N_b subsystems B_k, each integrating only smooth operations. The subsystems B_k are grouped together into clusters C_i if they are not separated by decision operations. The fixed point behavior of each C_i is modeled by a single noise source b_{gi}. The

statistical parameters of b_{gi} are calculated from the analytical model of cluster C_i.

The system obtained after the clustering phase forms a graph $G_{sys}(V_{sys}, E_{sys})$. The nodes set V_{sys} contains the clusters C_i and the operations O_i. The edges E_{sys} represent the signals connecting the operations and the clusters. In the considered example, the subsystems B_0 to B_3 are grouped together to form the cluster C_0. A noise source b_{g0} is added at the output of the cluster. Similarly, the subsystems B_5 and B_6 are grouped within the cluster C_1 and the noise source b_{g1} is associated with this cluster. The subsystem B_4 forms the cluster C_2.

In this hybrid approach, an analytical model is associated to each cluster. The objective of the hybrid approach is to obtain, the system output values for the N_p samples of each input I_m. For each input sample $I_m(n)$, the entire graph is covered. At each iteration of the optimization process, each decision operation O_i is processed in two steps. First, for each decision operation O_i, the possibility of a decision error is analyzed by studying the value of each input sample and the boundaries of the noise associated with this input. If the input value is far enough from the decision boundaries with respect to the limits of the noise, then there is no decision error. This condition is satisfied if the definition domain of the sum of the noise and the considered input value $x_i(n)$ is included in the decision region containing the input. If $x_i(n)$ is susceptible to generate a decision error, a random value is generated from a random process corresponding to the SNS model of the noise $b_{gi}(n)$. Then, the sum $x_i(n) + b_{gi}(n)$ is evaluated. If $b_{gi}(n)$ have generated a decision error, the rest of the graph is performed by simulation.

6.3 Framework for quality evaluation in presence of decision errors

Given the importance of the hybrid techniques and its ability to significantly accelerate the performance evaluation by reducing the number of simulations in comparison with simulation-based techniques, the proposed approach presented in 6.3 is based on the same concept : the system is simulated only when decision errors occur. It uses the system modelling proposed in [21] and described in the previous section. Our approach allows automating the quality evaluation by using the selective simulation framework described in Chapter 3. In this section, the specificities of the selective simulation framework, presented in chapter 3, are presented for the case of decision error analysis. For fractional word-length optimization, the proposed framework handles decision errors. The framework is applied to detect the indexes of decision error and simulate the application only when a decision error occurs. Subsequently, the steps T_1, T_2 and T_3 are detailed for the decision error analysis context.

6.3.1 Index classification (T_1)

The Index Classification step is illustrated in Figure 6.3. This step decides whether an error can occur at the output of the decision operation considering the reference signal at the input of the operation. In the first step, index classification, a reference floating point simulation of the entire system is performed. Then, for each decision operation O_i and for each input sample n, the possibility of a decision error is analyzed by studying the value $x_i(n)$ of the input x_i and the

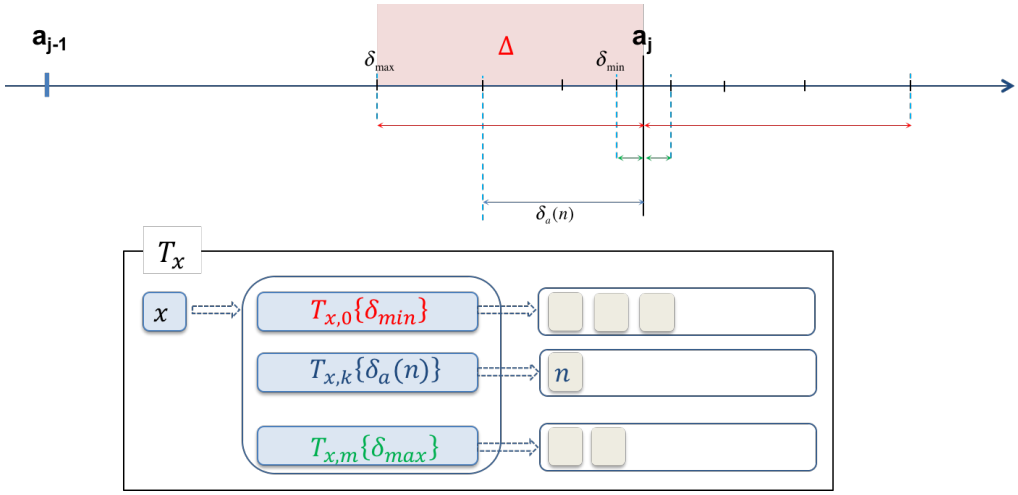


Figure 6.3 – Index Classification step for decision error detection.

boundaries of the noise b_{xi} associated with this input. The index n corresponds to a potential error If the value $x_i(n)$ is close enough to one of the decision boundaries taking into account the limits of the noise. This condition, is satisfied if $x_i(n)$ is in the domain Δ corresponding to the interval of the noise b_{xi} in the worst case as shown in Figure 6.3. This is evaluated by calculating the distance δ_a between the value of $x_i(n)$ and the decision boundaries a_j. Let δ_{min} be the minimal distance between the value of x and all the decision boundaries a_j associated to the decision operation O_i

$$\delta_a = \min_j \left(\delta_{a_j}\right) \tag{6.2}$$

Let δ_{max} be the maximal distance for which the value $x_i(n)$ is still in Δ. The index n corresponds to a potential error if the following condition is satisfied:

$$\delta_a < \delta_{max} \tag{6.3}$$

If δ_a is greater than this threshold, the decision error probability is considered to be null . This term δ_{max} is the parameter set by the developer and depends on the quantization noise bounds in the worst case.

The algorithm for the evaluation in this step is provided in Algorithm 5. First the reference samples corresponding to the index under consideration are obtained from the data-base. Then the distance δ_n of each value $x(n)$ from the different decision boundaries a_j is evaluated, if the value of $\delta_a(n)$ is in the interval $[0, \delta_{max}]$, then the index n is susceptible to decision error and it is saved in the structure $\mathcal{T}_x$ as shown in Figure 6.3.

The indexes n are classified and stored in the table according to their distance. This distance is expressed in terms of number of bits of precision. The rank k to access to the table $\mathcal{T}_x$ is determined with the following expression

```
class SigDec{
private:
  float w_min;
  float w_max;
  float value;
  ContextDec& _context; /* Link to the structure T and simulation context */
public:
   SigDec(float _value, float _w_max, float _w_min, ContextDec& _Context);
   void Save_Index(const double& a) const;
   ...
   friend bool operator < (const double& a, const SigDec& x);
   friend bool operator > (const double& a, const SigDec& x);
    friend bool operator <= (const double& a, const SigDec& x);
    friend bool operator >=  (const double& a, const SigDec& x);
    ...
    };
```

Listing 6.1 – Definition of SigDec C++ class

$$k = \max\left(\lfloor \log_2(\delta_n) \rfloor - \lfloor \log_2(\delta_{min}) \rfloor, 0\right) \tag{6.4}$$

where δ_{min} is the minimal considered distance. Thus all the distance lower than δ_{min} are stored in the same vector $\mathcal{T}_{x,0}$

Algorithm 5 Save eventual error index in Index Classification

Input: x(n) : input sample

$\delta_n = \min_j (|x(n) - a_j|)$

if $\delta_n \leq \delta_{min}$ **then**

$\mathcal{T}_{x,0,k_0} \leftarrow n$

$k_0 + +$ // Incrementation of the list index k_0

else

if $\delta_n < \delta_{max}$ **then**

$j = \max\left(\lfloor \log_2(\delta_n) \rfloor - \lfloor \log_2(\delta_{min}) \rfloor, 0\right)$

$\mathcal{T}_{x,j,k_j} \leftarrow n$

$k_j + +$ // Incrementation of the list index k_j

end if

end if

6.3.1.1 Implementation with SigDec C++ class

To simulate the application performance in presence of un-smooth error, two new classes are developed: Class `SigDec` allows determining the input indexes of un-smooth errors, and Class `SimDec`, described in Section 6.3.3, simulates the system at these indexes.

Currently, our approach is developed for relational operators (" $>$ "," $>=$ "," $<$ "," $<=$ ") and thus we support comparison with a threshold.

The class **SigDec** has several private members, as shown in the Listing 6.1. The variables w_{min} and w_{max} specify the limits of FWL for which the value is considered near the decision boundaries, i.e. may lead to an error. $value$ is the floating value of the variable. A link to the structure $\mathcal{T}$ is available.

The class overloads relational operators (" $>$ "," $>=$ "," $<$ "," $<=$ "). Therefore, any `SigDec` variable can be compared to floating-point variables and constants. The type `SigDec` is attributed to the floating variables involved in the comparison expression. The variables precision are collected during the simulation and the simulated index is stored in the suitable rank in the structure $\mathcal{T}$ as shown in Algorithm 5.

The operator overloading do not change the original value. It evaluates the difference between the floating value and the decision border, then calculate the number of bits of the difference to determine the suitable rank of the index using the Equation 6.4.

6.3.2 Index selection (T_2)

After saving the index of potential errors in step T_1, step T_2, illustrated in Figure 6.4, determines if an error occurs according to the tested word-lengths. Index Selection step save the time of checking every input samples if it leads to a decision error or not. The deviation d of the quantization noise at the input of the decision operation O_i can be determined from the tested word-length with an analytical approach for smooth sub-systems. d represents the maximum distance allowed from the boundary under which a decision error occurs. Any value $x_i(n)$ with a distance δ_a lower than d_i will lead to a decision error. The rank k_d , equal to $\lfloor \log_2(d) \rfloor$, is the maximal rank tolerated. All the index n in $\mathcal{T}$ of rank k lower than k_d lead to a decision error and are saved in the simulation list L_{gen}. This process is descried in Algorithm 6.

Algorithm 6 Detect error index in Index Selection

Input: x(n) : input sample

$k_d = \lfloor \log_2(d) \rfloor$

for ALL $k < k_d$ **do**

$L_{gen} \leftarrow L_{gen} \cup \mathcal{T}_{x,j}$

end for

The error propagated from previous block(s) are evaluated at this stage and the corresponding indexes are stored in the list L_{propag}. The propagated list are added to the simulation list L_{sim}

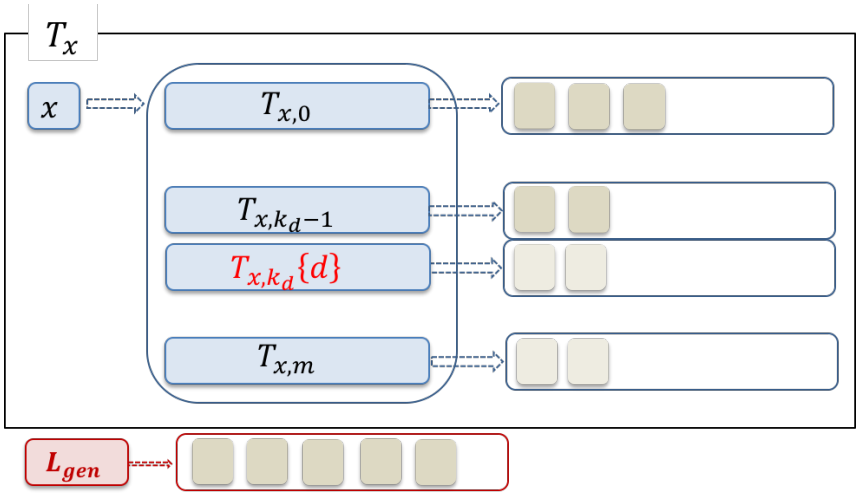


Figure 6.4 – Index Selection step for decision error detection.

$$L_{sim} \leftarrow L_{gen} \cup L_{propag} \tag{6.5}$$

6.3.3 Selective simulation (T_3)

The nominal reference value of the signal would be perturbed by quantization noise due to smooth fixed-point quantization. A random value is generated from a random process to mimic the SNS model and assigned to b_{xi}, the deviation of the noise depends on the precision allocated to the variable x_i . If the perturbation is large enough to generate an error at the output of the decision operation, then the system is simulated and SimMode is set to true. Otherwise, It is set to false. If a decision error occurs at output of O_i, all the blocks using a result resulting from the output of O_i must be processed by simulation.The simulation is performed in floating point and the fixed-point behavior of the blocks representing the simulated cluster is modeled by adding a noise source. This makes it possible to avoid performing fixed-point simulations, which are always more expensive in terms of execution time. Selective simulation is executed at each quality evaluation after defining a specific fractional word-length for each variable x_i. The process is described in Procedure 7.

Unlike the hybrid technique proposed in [21], our approach does not require to check each decision operation and at each quality evaluation if it leads to decision error or not. The system is browsed once in Index classification step. Then, at each quality evaluation only the sturcture $\mathcal{T}$ is browsed.

6.3.3.1 Implementation with SigDec C++ class

A new data class $SimDec$ have been developed to simulate the effect of decision error faster than traditional fixed-point data types. The declaration part of the class is given in the Listing 6.2.

Algorithm 7 Selective Simulation

$x(n) = GetReferenceSample(n)$
if $n \in L_{Sim}$ **then**
  $SimMode = TRUE$
  for ALL SMOOTH operation **do**
    $OUT_{SMOOTH} = OUT_{DATABASE}$
  end for
else
  $SimMode = FALSE$
  $OUT_{SYSTEM} = OUT_{DATABASE}$
end if

```
1  class SimDec{
2  private:
3    float high_E;
4    float low_E;
5    unsigned int accepted_deviation;
6    vector<my_index> Lsim;
7    ...
8  public:
9     SimDec(float _High_E, float _low_E, ContextDec& _context);
10    void Lsim_generation();
11    vector<my_index> get_Lsim();
12    ...
13 };
```

Listing 6.2 – Definition of SimDec C++ class

The variable *accepted_deviation* determines the maximum distance from the decision boundary under which an error occurs. All the indexes for which any variable leads to an error is stored in the simulation list `Lsim`. The class contains a method to generate `Lsim`.

6.4 Experiments

6.4.1 Edge detection application

To show the effectiveness of the proposed approach, the results obtained by applying our approach on a synthetic example are presented. This approach aims at accelerating the quality evaluation process compared to the traditional approach based on fixed-point simulation. Thus, the presented results show the output equivalence between the proposed technique and fixed-point

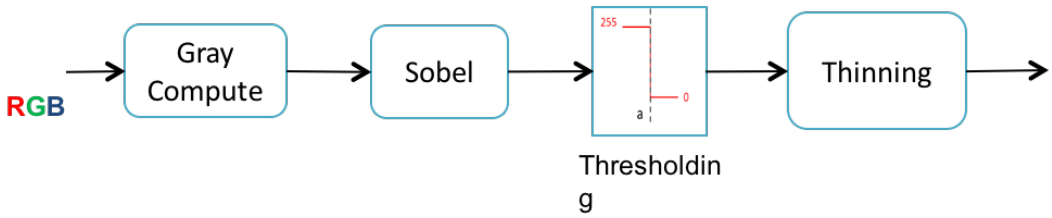


Figure 6.5 – Edge detection application considered to evaluate the efficiency of the proposed approach to eveluate the effect of decision error.

simulation and the execution time acceleration. In addition, the simulation time of our proposed technique is compared with that of the hybrid technique [6] and SNS model [110].

The application is a synthetic example that detects and tidy up the edge of an image by applying an edge detector and thinning operator. The flow graph of the considered application is shown in figure 6.5. The input is a color image defined by the three channel RGB (Red, Blue, Green). The color image is transformed into a gray image by the *ComputeGray* block to work on a single channel. Then, *Sobel* edge detector [111] followed by thresholding and morphological *Thinning* are applied successively on the gray image in order to identify the edges. Identifying edges in the image helps in object identification, motion tracking and image enhancement. It is also often performed on blurred images in order to sharpen or restore image quality [112].

The Sobel edge detector computes an approximation of the gradient in the vertical and horizontal direction. Thresholding aims at extracting the edges from the gradient values. This process leads to a binary image where white pixel represent the edges. The threshold is defined by the user and aims at selecting the pixels corresponding to the edges.

Thinning is used to remove selected foreground pixels from binary images. This morphological operation can be used for skeletonization. In this mode, it is commonly used to tidy up the output of edge detectors by reducing all lines to single pixel thickness. The *ComputeGray* and *Sobel* blocks are smooth while thresholding and *Thinning* blocks are un-smooth. Nevertheless, given that the *thinning* block process binary images, *i.e.* images whose pixels have only two possible intensity values and produces another binary image as output. Hence, a reduction of the number of bit can not be investigated. Thus, the *thinning* block does not generate un-smooth errors. Figure 6.6 shows the output of the four blocks for the Lena image. For a better visualization of the images, the color of the thresholding and thinning outputs are reversed. Black pixel represent the edges.

6.4.2 Experimental setup

A double precision simulation is carried out and the output of the blocks are stored. These results are considered as the reference. Using selective simulation approach, if an error occurred at the output of the threshold, the *Thinning* block has to be carried out only on the errors indexes. That will result in saving processing time when the threshold values are the same as the values

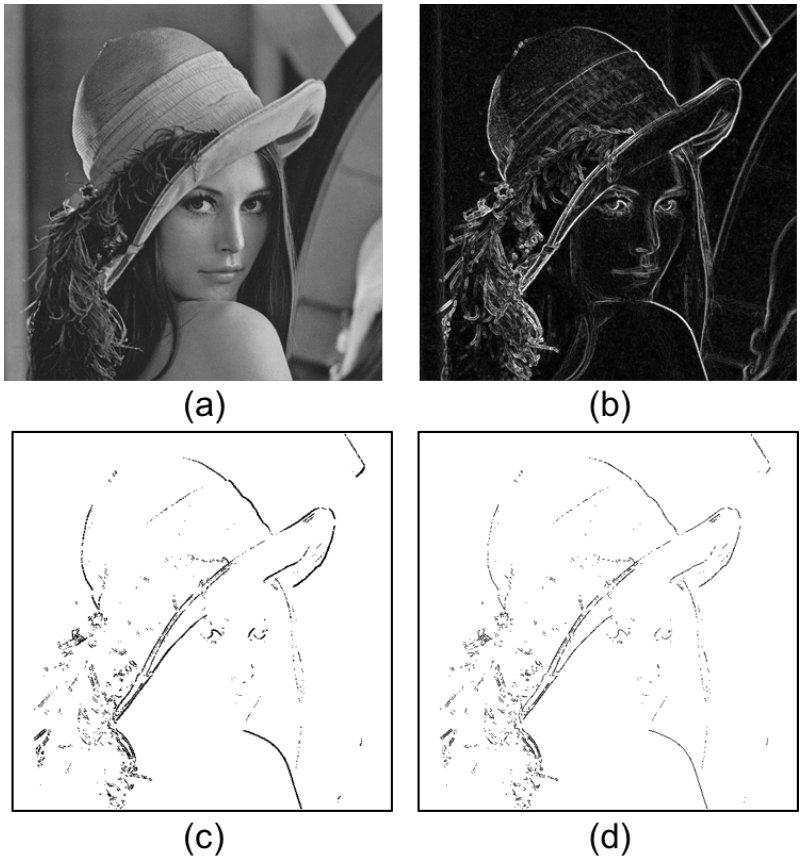


Figure 6.6 – Output of the different blocks of the edge detection application for the Lena image. a) Output of the *ComputeGray* image, b) Output of the *Sobel* block, c) Output of the thresholding operation, d) Output of the *Thinning* block.

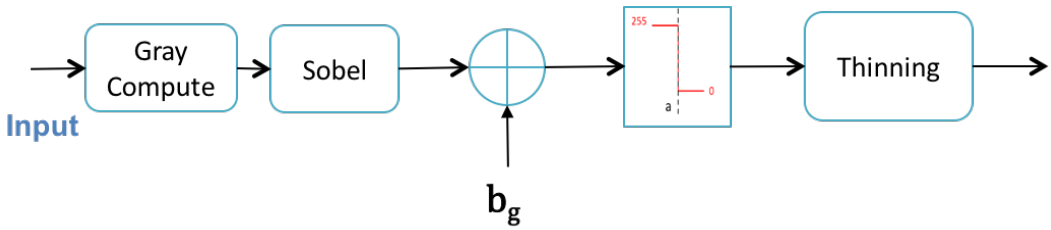


Figure 6.7 – Edge detection application using single noise source model.

obtained with reference simulation.

The tested image is Lena from the standard image processing test suite. The input image is filtered using a Gaussian low-pass filter to emulate the blurring effect. The edges of the blurred image are detected using the example shown in Figure 6.5.

The effect of finite precision due to fixed-point arithmetic is modelled with the SNS model as shown in Figure 6.7. A noise source is inserted at the input of the thresholding operation to emulate

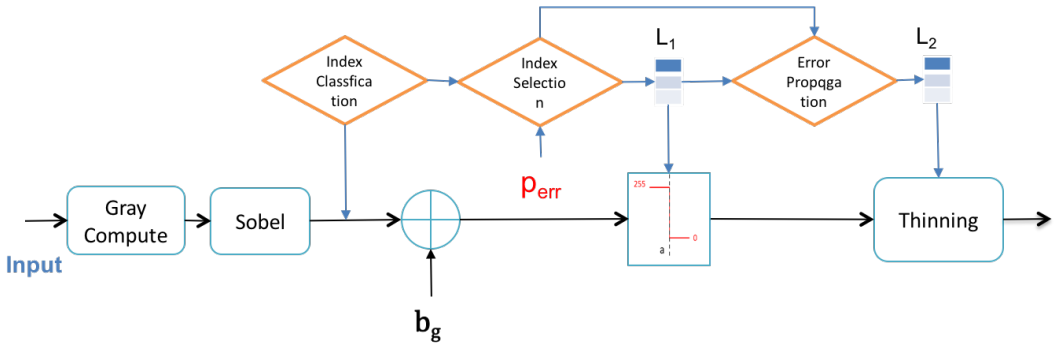


Figure 6.8 – Edge detection application using proposed techniquel.

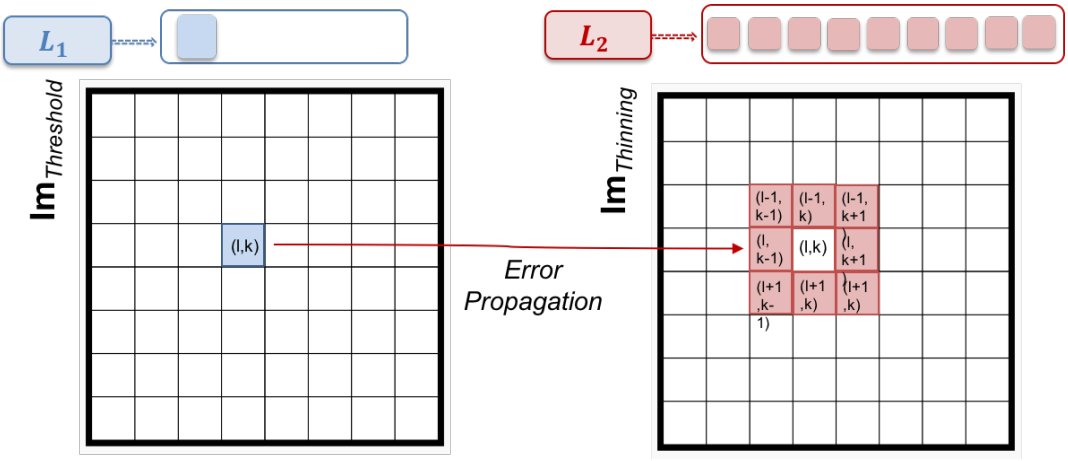


Figure 6.9 – Indexes of L_2 evaluated from L_1.

the quantization noise generated by the *ComputeGray* and *Sobel* blocks. A white Gaussian noise is considered.

The proposed technique is applied as shown in Figure 6.8. After simulating the first two blocks using infinite precision, a quantization noise is added to the output of the *Sobel* block to emulate the effect of fixed-point computation. The threshold and thinning blocks are simulated one time for index classification.

According to the tested word-length w, the bounds of the noise are computed and the deviation d is deducted. Index selection selects all the indexes leading to a potential error and store it in the simulation list. L_1 contains all the indexes for which the threshold operation should be simulated and L_2 contains the indexes of error propagating and leading to an error in the thinning block. The quality of the output is evaluated by simulating the threshold and thinning block for the indexed stored in L_1 and L_2 respectively. The indexes in L_2 are evaluated from L_1 as shown in Figure 6.9.

6.4.2.1 Quality metric

To analyze the effect of error decision on the edge detector application, the image obtained when decision errors occur and the reference image are compared and the Matthews correlation coefficient (MCC) is computed. The MCC is well suited to analyze the quality of binary classification *i.e.* classification in two classes C_P and C_N. It is a correlation coefficient between the observed and predicted binary classifications ($Predicted/Actual$). This metric takes into account the number of true positives T_P (C_P/C_P), the number of true negatives T_N (C_N/C_N), the number of false positives F_P (C_P/C_N) and the number of false negatives F_N (C_N/C_P). The MCC is computed with the following formula.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{6.6}$$

The MCC represents the correlation between the reference and predicted binary classifications. A coefficient equal to 1 means a perfect prediction and a value of 0 means that the prediction is equivalent to a random prediction.

6.4.3 Results of the experiments

6.4.3.1 Quality criteria

Figure 6.10 shows the output obtained by both proposed approach and fixed-point simulation for computation carried-out on 8-bit data. By comparing the image at the output of the two approaches, the two images are very close to each other. The difference between the two images is also shown in the Figure 6.10. In order to have a better understanding, the plots in Figure 6.11 show the MCC between the output of both fixed-point and proposed approach simulations in comparison with the infinite precision simulation for different probabilities of the decision error occurring after thresholding. The results in both cases are of the same order with very little difference for various decision error probabilities. This validates the statistical equivalence of the proposed approach with fixed-point simulation in this experiment.

When the precision is decreased, the decision errors increase and thus the output of fixed-point simulation is more deteriorated. This can be observed by calculating the MCC between the outputs of both floating-point and fixed-point simulations as shown in Figure 6.11. The same as for an overflow, a decision error corresponds to a noise with high power. Thus, when the number of error is high the performance of the output image is unacceptable. The simulations in this experiment are conducted for a minimum MCC between fixed-point and floating point simulation outputs equal to 0.88 corresponding to a maximum decision error equal to 0.37%. Thus, in this experiment, decision error can be considered as rare event.

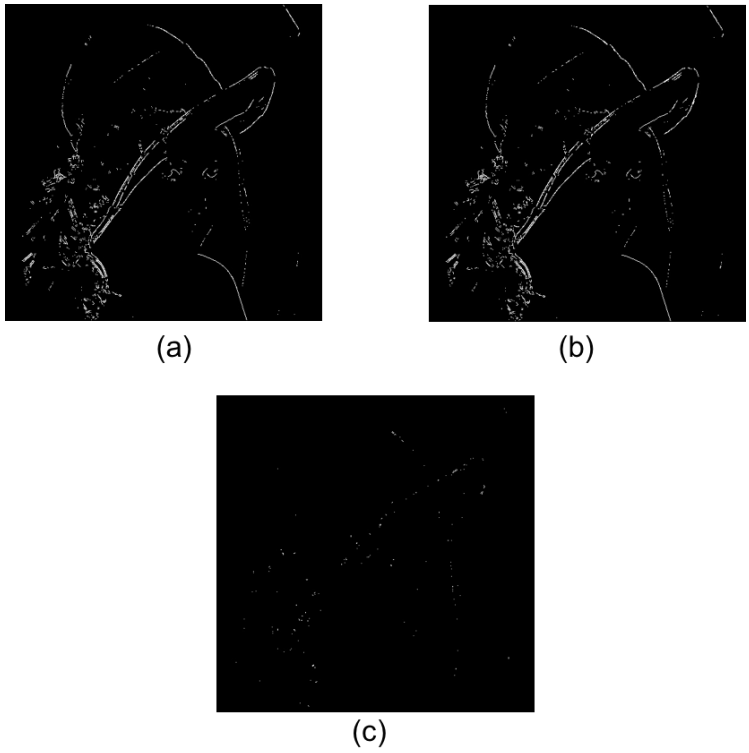


Figure 6.10 – Output of the thinning block for Lena image. a) Fixed-point simulation, b) Proposed approach simulation. c) Difference between the two images.

6.4.3.2 Execution time

The execution time of the quality evaluation process is evaluated using fixed-point simulation and the proposed approach for different computation precision (word-length w). Figure 6.12 shows the gain in time recorded when using our approach compared to the fixed-point simulation using Lena as input image. The gain in time is presented with respect to the decision error occurrence. As the number of bits increases, the decision error decreases and the image representation tends to be closer to the reference one. Therefore, the input indexes leading to decision error and needed to be simulated to evaluate the quality of the application decrease. Thus, the gain in time increases by several orders of magnitude with respect to the fixed-point simulation. In this experiment, the results lead to an acceleration up to 1000 .

In the word-length optimization process, several iterations are needed to obtain the optimized solution. To evaluate the effectiveness of the proposed approach, a steepest descent greddy algorithm (max-1 bit) is applied in order to find an optimized precision of the threshold block input image. To simplify, a uniform word-length strategy is used. Thus, the optimization process is made-up of one variable. The performance criteria evaluated is the MCC (M) between the resulted output and floating point simulation output. The algorithm, shown in Algorithm 8 starts with an initial word-length $w_0 = 16bits$. Then, the word-length w of the threshold block input image is reduced at each iteration while satisfying the performance criterion *i.e* M is higher or

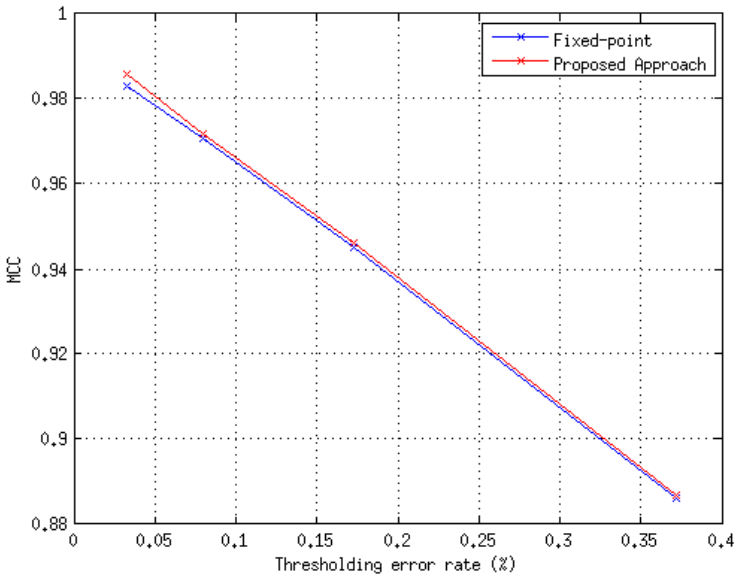


Figure 6.11 – MCC between both the proposed and fixed-point simulations output and floating point simulation output.

equal to the targeted MCC (M_{min}). Starting from w_0, the quality of the output image is evaluated at each iteration. When the target MCC is reached, the iterations stop and the precision w is allocated to the variable.

Algorithm 8 Optimization algorithm

$w \leftarrow w_0$
while $MCC > MCC_{min}$ **do**
  $w = w - 1$
end while
$w = w + 1$

Comparisons are conducted for fixed point simulation, SNS model, hybrid technique and the proposed approach. Several M_{min} are tested. Figure 6.13 shows the gain in terms of time obtained with the proposed approach, SNS model and the hybrid technique with respect to the fixed point simulation with respect to the target MCC. When M_{min} decreases, the number of iterations to obtain the optimal precision increases and the execution time grows. The gain obtained with respect to fixed point simulation increases with the number of iterations. Our approach records a gain up to 2000 for M_{min} equal to 0.88 given an output of MCC $M = 0.886$. While this gain reaches 110 using the SNS model and 1900 using the hybrid technique for the same MCC. Thus, our approach leads to a gain in time up to 90 with respect to SNS model and a gain between 1.1 and 1.5 with respect to the hybrid technique.

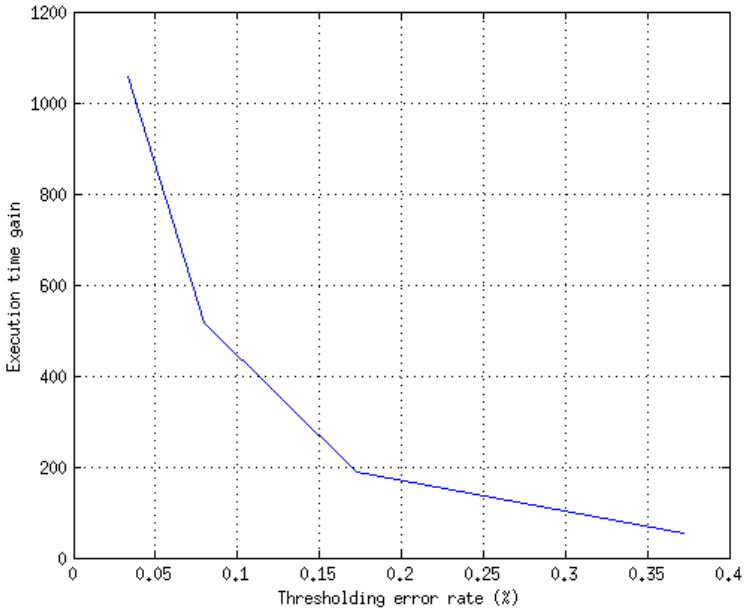


Figure 6.12 – Execution time Gain between the proposed approach and fixed point simulations with respect to decision error occurrence.

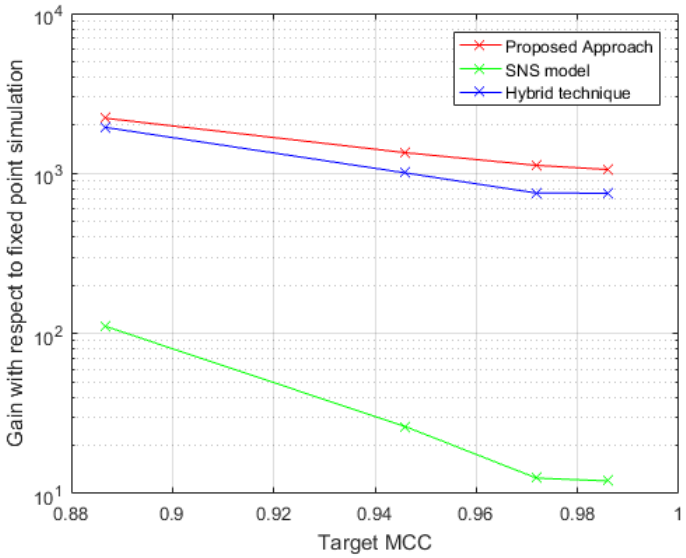


Figure 6.13 – Execution time with respect to MCC.

6.5 Conclusion

Decision error operations are un-smooth operations that cannot be described by analytical techniques. Simulation-based techniques have the advantage of being applied on all systems.

However, they leads to high execution time. The problem of evaluating fixed-point systems in presence of decision errors is presented in this chapter. After presenting the concept of un-smooth operations and the single noise source model, we explains the implementation of the framework proposed in Chapter 3 to accelerate the simulation of fixed-point system in presence of decision error. The proposed approach uses the SNS model to evaluate the impact of finite precision on smooth blocks while performing simulation of the decision error operations during fixed-point simulation. The approach is tested on an edge detector application and compared with fixed-point simulation, SNS model and an existing hybrid technique. The experimental results show significant acceleration of the simulations, where the execution time is reduced up to 1000 with respect to fixed-point simulation for one iteration. The gain in time increase when applying a max-1 bit optmization algorithm and reaches a value up to 2000 with respect to fixed-point simulation, up to 90 time with respect to SNS model and up to 1.5 times with respect to hybrid technique.

CHAPTER 7

Conclusions and Perspectives

In this thesis, we aim at accelerating the process of floating-point to fixed-point conversion. Floating-point and fixed-point operators are two popular choices available for the implementation of arithmetic operations. Fixed-point arithmetic operators are known to take significantly lesser area, shorter latency and are known to consume lesser power. Implementation of telecommunication algorithms usually have rigorous performance parameters to be achieved and demand high computational power. In such cases, using fixed-point arithmetic allows satisfying such constraints thanks to its ability to manipulate data with lower world-length compared to floating-point arithmetic. Thus, DSP algorithms are implemented into fixed-point architectures and floating-point to fixed-point conversion is mandatory.

In Chapter 2, floating-point and fixed-point representations are explained, and then compared in terms of arithmetic properties and hardware and software implementations. Then, the conversion from floating-point to fixed-point is explained. Methods used in the state of the art to determine the integer and fractional part world-lengths are presented.

Based on the state of art, the effects of overflow or un-smooth errors are not handled by analytical methods. These effects can be evaluated by traditional fixed-point simulation based approaches, but at the expense of long simulation time. Thus, a new framework that is based on selective simulation is presented in Chapter 3. We explained the problem of evaluating the system when an error (overflow or un-smooth error) occurs with low probability.

Then, we explained the concept of overflow and its relation with integer-world length, and the concept of unsmooth error and its relation with the quantization process. Moreover, a new approach that uses selective simulations to accelerate the simulation of finite precision effects analysis is presented. This approach can be applied on C based DSP applications using fixed-point arithmetic to evaluate the degradation due to overflow or un-smooth errors.

In Chapter 4, we detailed the proposed selective simulation approach when applied in the presence of overflow. After presenting the methodology of variable selection for integer world length optimization, the implementation of the proposed framework in C++ is explained. Then, experiments were conducted to verify the effectiveness of the proposed selective simulation technique. Results showed significant acceleration of the evaluation time. This evaluation time is reduced

up to 1000 time with respect to complete simulation based approach. These results show the effectiveness of the proposed approach.

Then, in Chapter 5, the proposed selective simulation approach is exploited to design a new algorithm for integer world-length optimization. This optimization problem is translated into the exploration of the trade-off between implementation cost and application quality. The cost and quality of systems implemented using fixed-point arithmetic depends on the assigned word-lengths. After presenting the existing algorithms to optimize the world-length, the proposed IWL optimization algorithm,which exploits the selective simulation technique used to evaluate the overflow effects on application performance, is detailed. This algorithm does not only reduce the cost, but also allows overcoming the long execution time of classical simulation based algorithms. Through experiments applied on the FFT part of an OFDM chain, the proposed algorithm results in a significant reduction of cost with acceptable degradation of quality criteria. At the same time, results show huge enhancement in the optimization time, where the acceleration factor reaches up to 617.

In Chapter 6, the proposed selective simulation framework is exploited in a hybrid technique to evaluate the performance of a fixed-point system containing decision operations. After presenting the methodology, experiments were conducted on an image processing application. The experimental results show significant acceleration of the simulations, where the execution time is reduced up to 1000 with respect to fixed-point simulation for one iteration. The gain in time increase when applying a max-1 bit optmization algorithm and reaches a value up to 2000 with respect to fixed-point simulation, up to 90 with respect to SNS model and up to 1.5 with respect to hybrid technique.

The work we have done in this thesis can be considered as a step towards the automation of fixed-point system design. The proposed approaches and algorithms have succeeded to overcome the limitation of simulation based approaches by accelerating its performance.

As a direct follow-up, the proposed approach should be tested on more systems to analyze its efficiency on different types of applications. The implementation of our approach does not consider recursive systems or systems with delays. It would be interesting to extend our implementation to suits such type of systems and test its performance on them.

The aim of the work done in this thesis is to enhance existing tools. This work can be integrated in the ID.fix tool, presented in Section 2.3 which is based on the compiler infrastructure GECOS. This will allow the conversion of the floating-point C source code into a fixed-point C-code. The GECOS tool can be used to automate the instrumentation of the C-source code.

CHAPTER 8

Résumé français

Contexte général

Au cours des dernières décennies, l'implémentation des applications de traitement numérique du signal a été facilitée par le progrès des technologies des semi-conducteurs. Le traitement numérique du signal permet de mettre en œuvre de nombreuses applications s'interfaçant avec le monde réel. Au niveau de l'industrie, la complexité grandissante de ces applications augmente la difficulté de la conception. Pour faire face à la concurrence entre les entreprises, ces dernières sont obligées de fournir plus de valeur pour chaque coût unitaire qu'elles facturent aux consommateurs. Pratiquement, les concepteurs doivent travailler dans un environnement de complexité croissante pour réduire le temps de mise sur le marché (*Time-to-Market*) et minimiser le coût du produit tout en satisfaisant les exigences du client.

Le secteur des télécommunication est l'une des industries en croissance rapide. Par exemple, le nombre d'abonnés aux réseaux mobiles (utilisateurs) a dépassé le nombre de 6 milliards dans le monde. L'industrie des télécommunications a bénéficié des progrès des technologies des semi-conducteurs. A titre d'exemple, les *smartphones* de nos jours intègrent une combinaison d'algorithmes de télécommunications, de traitement du signal et de la vidéo. L'implémentation de toutes ces applications doit répondre à la demande des utilisateurs en terme de qualité du résultat en sortie de l'application.

Le coût d'implémentation d'un appareil électronique moderne est habituellement évalué en termes de surface de silicium, de consommation d'énergie et de temps d'exécution. Ce coût doit être maintenu au minimum sans compromettre les performances du système. Ces objectifs sont contradictoires et le concepteur doit inévitablement décider d'un compromis entre la qualité et le coût. Par conséquent, il est très important de prendre des décisions pertinentes à chaque étape de la conception afin d'assurer la meilleure performance possible de l'ensemble du système. Une des décisions importantes est le choix de l'arithmétique utilisée pour mettre en œuvre ces algorithmes. Ce choix a un impact important sur le compromis entre le coût et la qualité. Les arithmétiques en virgule flottante et en virgule fixe sont les deux alternatives les plus communément utilisées. Les opérateurs utilisant l'arithmétique en virgule fixe conduisent à une surface, une latence et une consommation d'énergie nettement plus faible. Les algorithmes de télécommunication et de traitement du signal possèdent des exigences fortes en termes de performance et de consommation d'énergie. Dans de tels cas, l'utilisation de l'arithmétique virgule fixe permet de satisfaire

ces contraintes grâce à sa capacité à manipuler des données de taille plus faible par rapport à l'arithmétique en virgule flottante. Ainsi, les algorithmes de traitement numérique du signal sont implémentés au sein d'architectures en virgule fixe et une conversion de virgule flottante en virgule fixe est indispensable.

Le processus de conversion en virgule fixe est un problème d'optimisation ayant pour objectif la détermination de la largeur (nombre de bits) de chaque donnée. Ce problème d'optimisation permet d'explorer le compromis entre le coût et la qualité du résultat en sortie de l'application. En outre, le processus de conversion en virgule fixe est long. Celui-ci peut prendre jusqu'à 30% du temps de développement total. Ainsi, un outil de conversion automatique de virgule flottante en virgule fixe est essentiel pour optimiser efficacement le coût de l'implémentation et réduire les temps de développement. Depuis quelques années, des outils efficaces de synthèse de haut niveau sont disponibles. Ces outils génèrent des architectures au niveau RTL directement à partir d'une spécification de l'application en C ou C ++ et utilisant des types en virgule fixe. Ces outils réduisent considérablement le temps de développement tout en permettant une bonne exploration de l'espace de conception. Ainsi, la conversion en virgule fixe devient le goulet d'étranglement du développement rapide des produits.

Le processus de conversion de virgule flottante en virgule fixe est composé de deux parties correspondant à la détermination du nombre de bits pour la partie entière et du nombre de bits pour la partie fractionnaire. Pour réduire la complexité de la conversion, les optimisations de la partie entière et de la partie fractionnaire sont traitées séparément. L'analyse de la précision des calculs est liée à la notion de bruit de quantification et correspond à l'étude d'une métrique de qualité de l'application traduisant la sensibilité de la sortie à la présence de faibles perturbations liées à ces bruits de quantification. De nombreux travaux de recherche se sont concentrés sur l'optimisation de la partie fractionnaire en utilisant la puissance du bruit de quantification comme critère de qualité. Le nombre de bits minimal est un compromis entre le coût de l'implantation et la précision des calculs réalisés.

Description de notre travail

La première étape du processus de conversion en virgule fixe correspond à l'évaluation de la dynamique des différentes données. Cette évaluation de la dynamique permet de déterminer le nombre de bits minimal pour la partie entière d'une donnée à partir de ses valeurs maximales et minimales. Pour éviter l'apparition de débordements, les méthodes classiques d'évaluation de la dynamique calculent des limites théoriques absolues qui ne seront jamais dépassées dans la pratique. Ces méthodes fournissent des estimations pessimistes ce qui aboutit à une augmentation du coût de l'implantation. Comme l'absence de débordements est garantie, l'optimisation de la partie entière sous contrainte de qualité devient impossible et le compromis précision-coût d'implantation est exploré uniquement pour la partie fractionnaire.

Dans de nombreuses applications, des débordements occasionnels sont acceptables si leur probabilité d'occurrence est suffisamment faible. La méthode d'évaluation de la dynamique doit être en mesure de prendre en compte cette propriété. Des méthodes comme l'arithmétique

d'intervalle et l'arithmétique affine ne fournissent pas d'informations supplémentaires sur la répartition du signal dans l'intervalle de définition. Les signaux ayant de grandes variations et une faible probabilité pour les extremums ne sont pas bien pris en compte par les techniques basées sur l'arithmétique d'intervalle ou affine.

Différentes techniques ont été proposées pour estimer la fonction de densité de probabilité de tout type de données. Elles permettent de déterminer la dynamique pour une probabilité d'occurrence de débordement donnée. Ces techniques sont basées sur la théorie des Valeurs Extrêmes [15, 16, 17] ou des approches stochastiques comme la décomposition de Karhumen-Loeve (KLE) [18, 19] et la décomposition polynomial du Chaos (PCE) [20].

Les techniques stochastiques permettent d'estimer la FDP des données du système pour en déduire ensuite la dynamique des données. Dans les systèmes linéaires invariant dans le temps (*LIT*), la décomposition de Karhumen-Loeve (*KLE*) permet une discrétisation stochastique des données. En utilisant les propriétés de superposition des systèmes LIT, il est possible de déterminer la décomposition associée à la sortie. Cette technique prend en compte la corrélation temporelle et spatiale permettant d'améliorer l'estimation de la dynamique. La décomposition polynomial du Chaos (*PCE*) est utilisée pour les systèmes non LIT. Par rapport au KLE, une gamme plus étendue de systèmes est supportée, mais au détriment d'une plus grande complexité de calcul. Par ailleurs, la théorie des valeurs extrêmes est utilisée pour estimer la queue de la distribution associée à une donnée à partir d'un ensemble de simulations.

En général, ces techniques relient la probabilité de débordement et la dynamique. Cependant, la définition d'une expression mathématique des critères de qualité de l'application complique l'automatisation de la conception des systèmes en virgule fixe. D'une part, chaque application possède ses propres critères de qualité, tels que, par exemple, le taux d'erreur binaire pour le système de communication ou le MOS (Mean Opinion Score) pour les applications audio. D'autre part, une expression liant les critères de qualité de l'application et la probabilité de débordement de chaque donnée est difficile à établir.

L'erreur associée à la quantification d'une donnée est modélisée comme un bruit blanc additif et de distribution uniforme. Au niveau système, le modèle de source de bruit unique (*SNS*) permet de modéliser le bruit de quantification en sortie d'un sous-système comme la somme des différents bruit de quantification.

Les techniques basées sur les modèles de Widrow et la théorie de la perturbation ne sont valides que lorsque le pas de quantification est très petit par rapport à la dynamique. Au fur et à mesure que le pas de quantification augmente, les propriétés du bruit de quantification s'éloignent des prédictions analytiques et deviennent rapidement intraitables. Tels quantificateurs sont désignés sous le nom de quantificateurs non lisses et peuvent conduire à la présence d'erreurs de décision.

En outre, les méthodes existantes d'analyse de la précision numérique évaluent uniquement la puissance du bruit de quantification en sortie. Dans certains cas, comme l'évaluation de la qualité dans des systèmes avec des opérateurs de décision, cette information limitée s'avère insuffisante et la fonction de densité de probabilité totale du bruit doit être utilisée.

Les approches basées sur la simulation sont une alternative. Elles sont utilisées pour surmon-

ter la limitation des approches classiques. En comparaison avec les approches stochastiques, où l'établissement du lien entre les critères de qualité de l'application et la probabilité d'occurrence d'erreur n'est pas trivial dans le cas général, une approche basée sur la simulation peut être exécutée sur tout type de système [8, 23, 24] et permettent facilement d'évaluer la métrique de qualité. Cependant, ces méthodes prennent du temps et nécessitent un grand nombre d'échantillons pour obtenir une analyse précise. Il en résulte une sérieuse limitation de l'applicabilité des méthodes basées sur la simulation.

Les effets des débordements ou des erreurs de décision peuvent être évalués par des approches de simulation en virgule fixe. Ces approches peuvent être appliquées à tout type de système. Mais ces approches deviennent moins favorables en raison de leur long temps d'exécution. Néanmoins, les approches de simulation en virgule fixe sont exhaustives lorsque tous les échantillons d'entrée sont simulés pour chaque évaluation de l'analyse des effets des débordements.

Dans cette thèse, nous visons à accélérer le processus d'évaluation de la métrique de qualité. Nous proposons un nouvel environnement logiciel (*framework*) utilisant des simulations sélectives pour accélérer la simulation des effets des débordements et des erreurs de décision. Cette approche peut être appliquée à toutes les applications de traitement du signal développées en langage C. Par rapport aux approches classiques basées sur la simulation en virgule fixe, où tous les échantillons d'entrée sont traités, l'approche proposée simule l'application uniquement en cas d'erreur. En effet, les dépassements et les erreurs de décision doivent être des événements rares pour maintenir la fonctionnalité du système. Par conséquent, la simulation sélective permet de réduire considérablement le temps requis pour évaluer les métriques de qualité des applications. La simulation sélective permet de réduire considérablement le temps requis pour évaluer les critères de qualité des applications

La majeure partie des travaux est basée sur l'optimisation de la partie fractionnaire, cependant, l'optimisation du nombre de bits pour la partie fractionnaire peut diminuer de manière significative le coût de l'implantation lorsqu'une légère dégradation de la qualité de l'application est acceptable. En effet, de nombreuses applications sont tolérantes aux débordements si la probabilité d'occurrence de ces débordements est suffisamment faible. Ainsi, nous exploitons ce *framework* avec un nouvel algorithme d'optimisation des largeurs de données pour accélérer l'optimisation de la partie entière. En outre, nous appliquons la simulation sélective pour évaluer la qualité au cours de l'optimisation de la partie fractionnaire pour des systèmes comportant des opérateurs de décision.

Nous avons débuté nos travaux en examinant les méthodes existantes pour l'évaluation de la qualité des systèmes en virgule fixe. Cela nous a permis de comprendre les approches proposées et connaître les avantages et inconvénients de chaque technique. Nous avons étudié le problème d'évaluation de la qualité en sortie d'un système en virgule fixe en présence d'erreurs rares et de grandes amplitudes (débordement et erreur de décision). A cet égard, nous avons proposé une approche basée sur la technique de simulation sélective pour accélérer la simulation de l'évaluation de la qualité des applications pendant le processus d'optimisation de la largeur des données. En plus, nous avons proposé et développé un *framework* pour cette approche.

Dans un second temps, nous avons détaillé l'approche de simulation sélective proposée et ex-

pliqué sa procédure lorsqu'elle est appliquée pour évaluer la qualité en présence de débordement. Nous avons développé le *framework* en C++ et vérifié l'efficacité de la technique proposée en effectuant des expérimentations sur des applications de traitement du signal.

Ensuite nous avons proposé un algorithme d'optimisation de la partie entière qui exploite l'approche de simulation sélective proposée. Cet algorithme permet, non seulement de minimiser le coût, mais aussi de réduire considérablement le temps d'exécution. Nous avons testé l'efficacité de cet algorithme à travers des expérimentations sur le module FFT d'une chaîne de transmission numérique OFDM.

Finalement, nous avons adapté le *framework* pour accélérer l'évaluation de la qualité au cours de l'optimisation de la partie fractionnaire pour les systèmes en virgule fixe comportant des opérateurs de décision. L'approche proposée utilise le modèle de source de bruit unique pour évaluer l'effet de la précision finie sur les opérations de décision tout en effectuant la simulation lorsqu'une erreur de décision est présente. Intégré dans un algorithme d'optimisation heuristique, nous avons testé l'efficacité de l'approche sur une application de détection des contours au sein d'une image. Les expérimentations ont montré une réduction significative du temps d'exécution par rapport à la simulation en virgule fixe complète.

Bibliography

[1] D. Menard. *Methodologie de compilation d'algorithmes de traitement du signal pour les processeurs en virgule fixe, sous contrainte de precision*. PhD thesis, Universite de Rennes I, Lannion, december 2002.

[2] Olivier Sentieys, Daniel Menard, David Novo, and Karthick Parashar. Fixed-point refinement, a guaranteed approach towards energy efficient computing, Mar 2015. Tutorial at IEEE/ACM Design Automation and Test in Europe (DATE'15).

[3] A. Banciu. *A Stochastic Approach For The Range Evaluation*. PhD thesis, University of Rennes, Feb. 2012.

[4] O Sentieys, D Menard, and N Simon. Id. fix: an eda tool for fixed-point refinement of embedded systems. *University Booth of the IEEE/ACM Conference on Design, Automation and Test in Europe (DATE)*, 2011.

[5] K. Parashar, R. Rocher, D. Menard, and O. Sentieys. Analytical Approach for Analyzing Quantization Noise Effects on Decision Operators. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1554–1557, Dallas, march 2010.

[6] K. Parashar, D. Menard, R. Rocher, O. Sentieys, D. Novo, and F. Catthoor. Fast Performance Evaluation of Fixed-Point Systems with Un-Smooth Operators. In *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, 11 2010.

[7] Marco Ajmone Marsan, Giuseppina Bucalo, Alfonso Di Caro, Michela Meo, and Yi Zhang. Towards zero grid electricity networking: Powering bss with renewable energy sources. In *2013 IEEE international conference on communications workshops (ICC)*, pages 596–601. IEEE, 2013.

[8] Mathworks. *System-Level Design Products for DSP and communications*, 2001.

[9] Bruno Pinçon. *Une introductiona Scilab*. IECN, 2000.

[10] C. Shi and R. Brodersen. An automated floating-point to fixed-point conversion methodology. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 529–532, Hong Kong, april 2003.

[11] M. Clark, M. Mulligan, D. Jackson, and D. Linebarger. Accelerating Fixed-Point Design for MB-OFDM UWB Systems. *CommsDesign*, january 2005.

[12] T. Hill. AccelDSP Synthesis Tool Floating-Point to Fixed-Point Conversion of MATLAB Algorithms Targeting FPGAs. White papers, Xilinx, april 2006.

[13] Electronic system level design. http://www.mentor.com/esl/.

[14] Stratus high-level synthesis. http://www.forteds.com/.

[15] Emre Özer, Andy P. Nisbet, and David Gregg. A stochastic bitwidth estimation technique for compact and low-power custom processors. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):34:1–34:30, May 2008.

[16] E. Ozer, A.P. Nisbet, and D. Gregg. Stochastic Bitwidth Approximation Using Extreme Value Theory for Customizable Processors. Technical report, Trinity College, Dublin, october 2003.

[17] A. Chapoutot, L.S. Didier, and F. Villers. Range estimation of floating-point variables in simulink models. In *Conference on Design and Architectures for Signal and Image Processing (DASIP)*, pages 1–8, 2012.

[18] B. Wu, J. Zhu, and F. Najm. An analytical approach for dynamic range estimation. In *Proc. ACM/IEEE Design Automation Conference (DAC)*, pages 472–477, San Diego, june 2004.

[19] A. Banciu, E. Casseau, D. Menard, and T. Michel. Stochastic modeling for floating-point to fixed-point conversion. In *Proc. IEEE International Workshop on Signal Processing Systems, (SIPS)*, Beirut, october 2011.

[20] B. Wu, J. Zhu, and F.N. Najm. Dynamic range estimation for nonlinear systems. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 660–667, 2004.

[21] Karthick Parashar. *System-level approaches for fixed-point refinement of signal processing algorithms*. PhD thesis, Université Rennes 1, 2012.

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

[23] Mentor Graphics. *Algorithmic C Data Types*. Mentor Graphics, version 1.3 edition, march 2008.

[24] Open SystemC Initiative. SystemC User's Guide (ver 2.0). Technical report, www.systemc.org, 2001.

[25] R Nehmeh, D Menard, A Banciu, T Michel, and R Rocher. A fast method for overflow effect analysis in fixed-point systems. In *Design and Architectures for Signal and Image Processing (DASIP), 2014 Conference on*. IEEE, 2014.

[26] Riham Nehmeh, Daniel Menard, Erwan Nogues, Andrei Banciu, Thierry Michel, and Romuald Rocher. Fast integer word-length optimization for fixed-point systems. *Journal of Signal Processing Systems*, 2015.

[27] Riham Nehmeh, Daniel Menard, Andrei Banciu, Thierry Michel, and Romuald Rocher. Integer word-length optimization for fixed-point systems. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2014.

[28] B. Widrow, I. Kollár, and M.-C. Liu. Statistical Theory of Quantization. *IEEE Trans. on Instrumentation and Measurement*, 45(2):353–61, Apr. 1996.

[29] B. Widrow and I. Kollár. *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge University Press, Cambridge, UK, 2008.

[30] K. Kalliojärvi and J. Astola. Roundoff Errors in Block-Floating Point Systems. *IEEE Transactions on Signal Processing*, 44(4):783–790, april 1996.

[31] et al Novo, David. Ultra low energy domain specific instruction-set processor for on-line surveillance. In *Application Specific Processors (SASP), 2010 IEEE 8th Symposium*. IEEE, 2010.

[32] Chun Hok Ho, Chi Wai Yu, Philip Leong, Wayne Luk, and Steven JE Wilton. Floating-point fpga: architecture and modeling. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17, 2009.

[33] Chi Wai Yu, Julien Lamoureux, Steven JE Wilton, Philip HW Leong, and Wayne Luk. The coarse-grained/fine-grained logic interface in fpgas with embedded floating-point arithmetic units. *International Journal of Reconfigurable Computing*, 2009.

[34] Florent De Dinechin, Cristian Klein, and Bogdan Pasca. Generating high-performance custom floating-point pipelines. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*. IEEE, 2009.

[35] James David Balfour. *Efficient embedded computing*. PhD thesis, Stanford University, 2010.

[36] R. Cmar, L. Rijnders, P. Schaumont, and I. Bolsens. A Methodology and Design Environment for DSP ASIC Fixed Point Refinement. In *Proc. IEEE/ACM conference on Design, Automation and Test in Europe (DATE)*, pages 271–276, Munich, march 1999.

[37] Seehyun Kim, Ki-Il Kum, and Wonyong Sung. Fixed-point optimization utility for C and C++ based digital signal processing programs. *IEEE Transactions on Circuits and Systems II - Analog and Digital Signal Processing*, 45(11):1455 –1464, nov 1998.

[38] K. Kum, J.Y. Kang, and W.Y. Sung. AUTOSCALER for C: An optimizing floating-point to integer C program converter for fixed-point digital signal processors. *IEEE Transactions on Circuits and Systems II - Analog and Digital Signal Processing*, 47(9):840–848, september 2000.

[39] R. Kearfott. Interval Computations: Introduction, Uses, and Resources. *Euromath Bulletin*, 1996.

[40] Carlos Carreras, Juan López, Octavio Nieto-Taladriz, et al. Bit-width selection for data-path implementations. In *System Synthesis, 1999. Proceedings. 12th International Symposium on*. IEEE, 1999.

[41] A Benedetti and Pietro Perona. Bit-width optimization for configurable dsp's by multi-interval analysis. In *Signals, Systems and Computers, 2000. Conference Record of the Thirty-Fourth Asilomar Conference on*, 2000.

[42] Jason Cong, Karthik Gururaj, Bin Liu, Chunyue Liu, Zhiru Zhang, Sheng Zhou, and Yi Zou. Evaluation of Static Analysis Techniques for Fixed-Point Precision Optimization. In *FCCM '09: Proceedings of the 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines*, pages 231–234, Washington, DC, USA, 2009. IEEE Computer Society.

[43] C. Fang, R. Rutenbar, M Püschel, and T Chen. Toward efficient static analysis of finite-precision effects in DSP applications via affine arithmetic modeling. In *Proc. ACM/IEEE Design Automation Conference (DAC)*, pages 496–501, New York, 2003.

[44] L.H. de Figueiredo and J. Stolfi. Affine arithmetic: Concepts and applications. *Numerical Algorithms*, 2004.

[45] Luiz H de Figueiredo. Self-validated numerical methods and applications. In *Brazilian Mathematics Colloquium monograph, IMPA, Rio de Janeiro, Brazil*, 1997.

[46] Joan Carletta, Robert Veillette, Frederick Krach, and Zhengwei Fang. Determining appropriate precisions for signals in fixed-point iir filters. In *Proceedings of the 40th annual Design Automation Conference*. ACM, 2003.

[47] Leland B Jackson. On the interaction of roundoff noise and dynamic range in digital filters. *Bell System Technical Journal*, 1970.

[48] A. Oppenheim and R.W. Schafer. *Discrete Time Signal Processing*. Prentice All Signal Processing series. Prentice All, Upper Saddle River, 2 edition, 1999.

[49] Emil Julius Gumbel. *Statistics of extremes*. Courier Corporation, 2012.

[50] L. Zhang, Y. Zhang, and W. Zhou. Floating-point to fixed-point transformation using extreme value theory. In *Eighth IEEE/ACIS International Conference on Computer and Information Science (ICIS)*, pages 271–276, 2009.

[51] Claire Fang Fang, Tsuhan Chen, Rob Rutenbar, et al. Floating-point error analysis based on affine arithmetic. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2. IEEE, 2003.

[52] J.-M. Chesneaux, L.-S. Didier, and F. Rico. Fixed CADNA library. In *Proc. conference on Real Number Conference (RNC)*, pages 215–221, Lyon, France, september 2003.

[53] S. Kim, , and W. Sung. Word-Length Optimization for High Level Synthesis of Digital Signal Processing Systems. In *IEEE Workshop on SiGNAL Processing Systems*, pages 142–151, Boston, october 1998.

[54] SystemC User's Guide. Version 2.0. *Earth Sciences*, 2001.

[55] Martin Coors, Holger Keding, Olaf Lüthje, and Heinrich Meyr. Integer code generation for the ti tms320c62x. In *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*, volume 2. IEEE, 2001.

[56] S. Kim and W. Sung. Fixed-Point Error Analysis and Word Length Optimization of 8x8 IDCT Architectures. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(8):935–940, december 1998.

[57] J. Kang and W. Sung. Fixed-Point C Compiler for TMS320C50 Digital Signal Processor. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Munich, april 1997.

[58] L. De Coster, M. Ade, R. Lauwereins, and J.A. Peperstraete. Code Generation for Compiled Bit-True Simulation of DSP Applications. In *Proc. IEEE International Symposium on System Synthesis (ISSS)*, pages 9–14, Hsinchu, december 1998.

[59] L. De Coster. *Bit-True Simulation of Digital Signal Processing Applications*. PhD thesis, KU Leuven, 1999.

[60] H. Keding, M. Willems, M. Coors, and H. Meyr. FRIDGE: A Fixed-Point Design and Simulation Environment. In *Proc. IEEE/ACM conference on Design, Automation and Test in Europe (DATE)*, pages 429–435, Paris, march 1998.

[61] George A. Constantinides. Word-length optimization for differentiable nonlinear systems. *ACM Transactions on Design Automation of Electronic Systems*, 11(1):26–43, 2006.

[62] R. Rocher, D. Menard, P. Scalart, and O. Sentieys. Analytical accuracy evaluation of Fixed-Point Systems. In *Proc. European Signal Processing Conference (EUSIPCO)*, Poznan, September 2007.

[63] J.A. Lopez, G. Caffarena, C. Carreras, and O. Nieto-Taladriz. Fast and accurate computation of the roundoff noise of linear time-invariant systems. *IET Circuits, Devices and Systems*, 2(4):393–408, august 2008.

[64] G. Caffarena, J.A. LÃşpez, . Fernandez, and C. Carreras. SQNR Estimation of Fixed-Point DSP Algorithms. *EURASIP Journal on Advance Signal Processing*, volume 2010, 2010.

[65] D. Menard, R. Rocher, and O. Sentieys. Analytical Fixed-Point Accuracy Evaluation in Linear Time-Invariant Systems. *IEEE Transactions on Circuits and Systems I: Regular Papers,*, 55(1), November 2008.

[66] R. Rocher, D. Ménard, and O. Sentieysand P. Scalart. Analytical Approach for Numerical Accuracy Estimation of Fixed-Point Systems Based on Smooth Operations. *IEEE Transactions on Circuits and Systems. Part I, Regular Papers*, 59(10):2326 – 2339, October 2012.

[67] C. Shi and R. Brodersen. A perturbation theory on statistical quantization effects in fixed-point DSP with non-stationary inputs. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 373–376, Vancouver, may 2004.

[68] P.D. Fiore. Efficient Approximate Wordlength Optimization. *IEEE Transactions on Computers*, 57(11):1561 –1570, november 2008.

[69] K. Kum and W. Sung. Combined Word-Length Optimization and High-level Synthesis of Digital Signal Processing Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20:921–930, August 2001.

[70] 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):3087–3090, Dec. 1995.

[71] Linsheng Zhang, Yan Zhang, and Wenbiao Zhou. Fast trade-off evaluation for digital signal processing systems during wordlength optimization. In *Proceedings of the 2009 International Conference on Computer-Aided Design*. ACM, 2009.

[72] Jorge Stol and Luiz Henrique De Figueiredo. Self-validated numerical methods and applications. In *Monograph for 21st Brazilian Mathematics Colloquium, IMPA, Rio de Janeiro*. Citeseer, 1997.

[73] J Lopez, Carlos Carreras, Gabriel Caffarena, Octavio Nieto-Taladriz, et al. Fast characterization of the noise bounds derived from coefficient and signal quantization. In *Circuits and*

Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on, volume 4. IEEE, 2003.

[74] S. Wadekar and A. Parker. Accuracy sensitive word-length selection for algorithm optimization . In *Proc. International Conference on Computer Design (ICCD)*, pages 54 –61, october 1998.

[75] Altaf Abdul Gaffar, Oskar Mencer, Wayne Luk, Peter YK Cheung, and Nabeel Shirazi. Floating-point bitwidth analysis via automatic differentiation. In *Field-Programmable Technology, 2002.(FPT). Proceedings. 2002 IEEE International Conference on*. IEEE, 2002.

[76] Unifying bit-width optimisation for fixed point and floating-point designs. Unifying bit-width optimisation for fixed-point and floating-point designs. In *Field-Programmable Custom Computing Machines, 2004. FCCM 2004. 12th Annual IEEE Symposium on*. IEEE, 2004.

[77] Yu Pang, O Sarbishei, K Radecka, and Zeljko Zilic. Challenges in verifying and optimizing fixed-point arithmetic-intensive designs. In *Automation Quality and Testing Robotics (AQTR), 2010 IEEE International Conference on*. IEEE, 2010.

[78] Yu Pang, Kartazyna Radecka, and Zeljko Zilic. Optimization of imprecise circuits represented by taylor series and real-valued polynomials. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(8), 2010.

[79] Omid Sarbishei, Katarzyna Radecka, and Zeljko Zilic. Analytical optimization of bit-widths in fixed-point lti systems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(3), 2012.

[80] Omid Sarbishei and Katarzyna Radecka. On the fixed-point accuracy analysis and optimization of polynomial specifications. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 32(6), 2013.

[81] Luis Esteban, Jesus Lopez, E Sedano, S Hernandez-Montero, Miriam Sanchez, et al. Quantization analysis of the infrared interferometer of the tj-ii stellarator for its optimized fpga-based implementation. *Nuclear Science, IEEE Transactions on*, 60(5), 2013.

[82] D. Menard, R. Serizel, R. Rocher, and O. Sentieys. Accuracy Constraint Determination in Fixed-Point System Design. *EURASIP Journal on Embedded Systems*, 2008:12, 2008.

[83] Romuald Rocher and Pascal Scalart. Noise probability density function in fixed-point systems based on smooth operators. In *Design and Architectures for Signal and Image Processing (DASIP), 2012 Conference on*. IEEE, 2012.

[84] C. Shi and R. Brodersen. Floating-point to fixed-point conversion with decision errors due to quantization. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Montreal, may 2004.

[85] Antoine Floc'h, Tomofumi Yuki, Ali El-Moussawi, Antoine Morvan, Ken Martin, Maxime Naullet, Mythri Alle, Ludovic L'Hours, Nicolas Simon, Steven Derrien, et al. Gecos: A framework for prototyping custom hardware design flows. In *Source Code Analysis and Manipulation (SCAM), 2013 IEEE 13th International Working Conference on*. IEEE, 2013.

[86] EPI Cairn. The gecos (generic compiler suite) source-to-source compiler infrastructure. *En ligne: http://gecos. gforge. inria. fr*.

[87] A. Morvan, S. Derrien, and P. Quinton. Polyhedral bubble insertion: A method to improve nested loop pipelining for high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(3):339–352, 2013.

[88] Yilin Zhao. Standardization of mobile phone positioning for 3g systems. *Communications Magazine, IEEE*, 40(7), 2002.

[89] Ralph Bucher and D Misra. A synthesizable vhdl model of the exact solution for three-dimensional hyperbolic positioning system. *Vlsi Design*, 15(2):507–520, 2002.

[90] J. Armstrong. Ofdm for optical communications. *Lightwave Technology*, 27(3):189 – 204, 2009.

[91] George A Constantinides and Gerhard J Woeginger. The complexity of multiple wordlength assignment. *Applied mathematics letters*, 15(2), 2002.

[92] Kyungtae Han, Iksu Eo, Kyungsti Kim, and Hanjin Cho. Numerical word-length optimization for cdma demodulator. In *Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on*. IEEE, 2001.

[93] William G Osborne, Ray CC Cheung, J Coutinho, Wayne Luk, and Oskar Mencer. Automatic accuracy-guaranteed bit-width optimization for fixed and floating-point systems. In *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*. IEEE, 2007.

[94] Arindam Mallik, Debjit Sinha, Prithu Banerjee, and Hai Zhou. Low-power optimization by smart bit-width allocation in a systemc-based asic design environment. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26, 2007.

[95] N. Herve. *Contributions a la synthese d'architecture virgule fixe a largeurs multiples*. PhD thesis, Universite de Rennes 1, Lannion, march 2007.

[96] Fred Glover. Tabu search-part ii. *ORSA Journal on computing*, 2, 1990.

[97] Fred Glover. Tabu search-part i. *ORSA Journal on computing*, 1, 1989.

[98] H.-N. Nguyen, D. Menard, and O. Sentieys. Novel Algorithms for Word-length Optimization. In *Proc. European Signal Processing Conference (EUSIPCO)*, Barcelona, september 2011.

[99] H Choi and WP Burleson. Search-based wordlength optimization for vlsi/dsp synthesis. In *VLSI Signal Processing, VII, 1994.,[Workshop on]*. IEEE, 1994.

[100] D. Menard, D. Chillet, and O. Sentieys. Floating-to-fixed-point Conversion for Digital Signal Processors. *EURASIP Journal on Applied Signal Processing*, 2006:1–19, january 2006.

[101] G. Constantinides, P. Cheung, and W. Luk. Wordlength optimization for linear digital signal processing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(10):1432– 1442, october 2003.

[102] G. A. Constantinides, P. Y. K. Cheung, and W. Luk. *Synthesis and Optimization of DSP Algorithms*. Kluwer Academic, 2004.

[103] F. Catthoor, H. de Man, and J. Vandewalle. Simulated-annealing-based optimization of coefficient and data word-lengths in digital filters. *International Journal of Circuit Theory and Applications*, I:371âĂŞ390, 1988.

[104] D.-U. Lee, A.A. Gaffar, R.C.C. Cheung, O. Mencer, W. Luk, and G.A. Constantinides. Accuracy-Guaranteed Bit-Width Optimization. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25(10):1990 –2000, 2006.

[105] R Nambiar, CKK Tang, and P Mars. Genetic and learning automata algorithms for adaptive digital filters. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 4. IEEE, 1992.

[106] N. Herve, D. Menard, and O. Sentieys. Data Wordlength Optimization for FPGA Synthesis. In *Proc. IEEE International Workshop on Signal Processing Systems, (SIPS)*, pages 623–628, Athens, november 2005.

[107] Brad S Chissom. Interpretation of the kurtosis statistic. *The American Statistician*, 1970.

[108] Changchun Shi. *Floating-point to Fixed-point Conversion*. PhD thesis, University of California, Berkeley, 2004.

[109] Aymen Chakhari. *Evaluation analytique de la précision des systèmes en virgule fixe pour des applications de communication numérique*. PhD thesis, Université Rennes 1, 2014.

[110] K. Parashar, R. Rocher, D. Menard, and O. Sentieys. A Hierarchical Methodology for Word-Length Optimization of Signal Processing Systems. In *Proc. International Conference on VLSI Design*, Bangalore, January 2010.

[111] Tamar Peli and David Malah. A study of edge detection algorithms. *Computer graphics and image processing*, 20, 1982.

[112] Frank Y Shih. *Image processing and mathematical morphology: fundamentals and applications*. 2009.

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Quality Evaluation in Fixed-point Systems with Selective Simulation

Nom Prénom de l'auteur : NEHMEH RIHAM

Membres du jury :
- Monsieur PAINDAVOINE Michel
- Monsieur LACASSAGNE Lionel
- Monsieur MENARD Daniel
- Madame NOUVEL Fabienne
- Monsieur MARTIN Eric
- Monsieur BANCIU Andrei

Président du jury : Eric Martin.

Date de la soutenance : 13 Juin 2017

Reproduction de la these soutenue

Thèse pouvant être reproduite en l'état
~~Thèse pouvant être reproduite après corrections suggérées.~~

Fait à Rennes, le 13 Juin 2017

Signature du président de jury

Le Directeur,

M'hamed DRISSI

Résumé

Le temps de mise sur le marché et les coûts d'implantation sont les deux critères principaux à prendre en compte dans l'automatisation du processus de conception de systèmes numériques. Les applications de traitement du signal utilisent majoritairement l'arithmétique virgule fixe en raison de leur coût d'implantation plus faible. Ainsi, une conversion en virgule fixe est nécessaire. Cette conversion est composée de deux parties correspondant à la détermination du nombre de bits pour la partie entière et pour la partie fractionnaire. Le raffinement d'un système en virgule fixe nécessite d'optimiser la largeur des données en vue de minimiser le coût d'implantation tout en évitant les débordements et un bruit de quantification excessif.

Les applications dans les domaines du traitement d'image et du signal sont tolérantes aux erreurs si leur probabilité ou leur amplitude est suffisamment faible. De nombreux travaux de recherche se concentrent sur l'optimisation de la largeur de la partie fractionnaire sous contrainte de précision. La réduction du nombre de bits pour la partie fractionnaire conduit à une erreur d'amplitude faible par rapport à celle du signal. La théorie de la perturbation peut être utilisée pour propager ces erreurs à l'intérieur des systèmes à l'exception du cas des opérations *un-smooth*, comme les opérations de décision, pour lesquelles une erreur faible en entrée peut conduire à une erreur importante en sortie.

De même, l'optimisation de la largeur de la partie entière peut réduire significativement le coût lorsque l'application est tolérante à une faible probabilité de débordement. Les débordements conduisent à une erreur d'amplitude élevée et leur occurrence doit donc être limitée. Pour l'optimisation des largeurs des données, le défi est d'évaluer efficacement l'effet des erreurs de débordement et de décision sur la métrique de qualité associée à l'application. L'amplitude élevée de l'erreur nécessite l'utilisation d'approches basées sur la simulation pour évaluer leurs effets sur la qualité.
Dans cette thèse, nous visons à accélérer le processus d'évaluation de la métrique de qualité. Nous proposons un nouvel environnement logiciel utilisant des simulations sélectives pour accélérer la simulation des effets des débordements et des erreurs de décision. Cette approche peut être appliquée à toutes les applications de traitement du signal développées en langage C. Par rapport aux approches classiques basées sur la simulation en virgule fixe, où tous les échantillons d'entrée sont traités, l'approche proposée simule l'application uniquement en cas d'erreur. En effet, les dépassements et les erreurs de décision doivent être des événements rares pour maintenir la fonctionnalité du système. Par conséquent, la simulation sélective permet de réduire considérablement le temps requis pour évaluer les métriques de qualité des applications.

De plus, nous avons travaillé sur l'optimisation de la largeur de la partie entière, qui peut diminuer considérablement le coût d'implantation lorsqu'une légère dégradation de la qualité de l'application est acceptable. Nous exploitons l'environnement logiciel proposé auparavant à travers un nouvel algorithme d'optimisation de la largeur des données. La combinaison de cet algorithme et de la technique de simulation sélective permet de réduire considérablement le temps d'optimisation.

Abstract

Time-to-market and implementation cost are high-priority considerations in the automation of digital hardware design. Nowadays, digital signal processing applications use fixed-point architectures due to their advantages in terms of implementation cost. Thus, floating-point to fixed-point conversion is mandatory. The conversion process consists of two parts corresponding to the determination of the integer part word-length and the fractional part world-length. The refinement of fixed-point systems requires optimizing data word-length to prevent overflows and excessive quantization noises while minimizing implementation cost.

Applications in image and signal processing domains are tolerant to errors if their probability or their amplitude is small enough. Numerous research works focus on optimizing the fractional part word-length under accuracy constraint. Reducing the number of bits for the fractional part word-length leads to a small error compared to the signal amplitude. Perturbation theory can be used to propagate these errors inside the systems except for unsmooth operations, like decision operations, for which a small error at the input can leads to a high error at the output.

Likewise, optimizing the integer part word-length can significantly reduce the cost when the application is tolerant to a low probability of overflow. Overflows lead to errors with high amplitude and thus their occurrence must be limited. For the word-length optimization, the challenge is to evaluate efficiently the effect of overflow and unsmooth errors on the application quality metric. The high amplitude of the error requires using simulation based-approach to evaluate their effects on the quality.

In this thesis, we aim at accelerating the process of quality metric evaluation. We propose a new framework using selective simulations to accelerate the simulation of overflow and un-smooth error effects. This approach can be applied on any C based digital signal processing applications. Compared to complete fixed-point simulation based approaches, where all the input samples are processed, the proposed approach simulates the application only when an error occurs. Indeed, overflows and unsmooth errors must be rare events to maintain the system functionality. Consequently, selective simulation allows reducing significantly the time required to evaluate the application quality metric.

Moreover, we focus on optimizing the integer part, which can significantly decrease the implementation cost when a slight degradation of the application quality is acceptable. Indeed, many applications are tolerant to overflows if the probability of overflow occurrence is low enough. Thus, we exploit the proposed framework in a new integer word-length optimization algorithm. The combination of the optimization algorithm and the selective simulation technique allows decreasing significantly the optimisation time.

INSA RENNES

N° d'ordre : **17ISAR 15 / D17 - 15**
Institut National des Sciences Appliquées de Rennes
20, Avenue des Buttes de Coësmes - CS 14315 - F-35043 Rennes Cedex
Tél : 02 23 23 82 00 – Fax : 02 23 23 83 96