Fixed-point refinement of digital signal processing systems

Daniel Menard, Gabriel Caffarena, Juan Antonio Lopez, David Novo, Olivier Sentieys

Fixed-Point Refinement of Digital Signal Processing Systems

D. Menard[*], G. Caffarena[†], J.A. Lopez[‡],
D. Novo[§], O. Sentieys[¶]

February 4, 2019

Contents

[*] Univ Rennes, INSA Rennes, IETR, Rennes, France
[†] University CEU-San Pablo, Madrid, Spain
[‡] ETSIT, Universidad Politécnica de Madrid, Spain
[§] LIRMM, Université de Montpellier, CNRS, Montpellier, France
[¶] Univ Rennes, Inria, Rennes, France

Abstract

For the digital signal processing (DSP) part of mixed-signal systems, fixed-point arithmetic is favoured due to its high efficiency in terms of implementation cost, namely energy consumption, area, and latency. Fixed-point arithmetic provides low-cost operators at the expense of an unavoidable error between the ideal precision values and the finite precision ones. With a careful data word-length optimisation process, the designer can safely profit from the inherent trade-off between implementation cost and computation accuracy. In this chapter, we describe in detail the different stages comprising the fixed-point conversion process (i.e., dynamic range evaluation, word-length optimisation, and accuracy evaluation) and propose practical solutions to address each stage.

1 Introduction

In the quest for lower power consumption and higher performance, mixed-signal applications progressively rely on digital signal processing. Designers are undertaking a paradigm shift from high performance analog circuits to digitally assisted analog circuits. Compared with the analog circuits, digital circuits are smaller, faster, more power efficient, and more flexible. Thus, the new trend is to build simple analog blocks followed by an increasingly complex digital post-processor that applies corrections to the output. Yet, the cost benefits of digital signal processing are not for free and can only be attained through an elaborated design methodology able to restrain the finite precision effects derived from limited bit-widths.

Digital systems are invariably subject to non-idealities derived from their finite precision arithmetic. A digital operator (e.g., an adder or a multiplier) imposes a limited number of bits (i.e., word-length) upon its inputs and outputs. As a result, the values produced by such an operator suffer from (small) deviations with respect to the values produced by its equivalent (infinite precision) mathematical operation (e.g., the addition or the multiplication). The more the bits allocated the smaller the deviation (or quantisation error) but also the larger, the slower and the more energy hungry the operator.

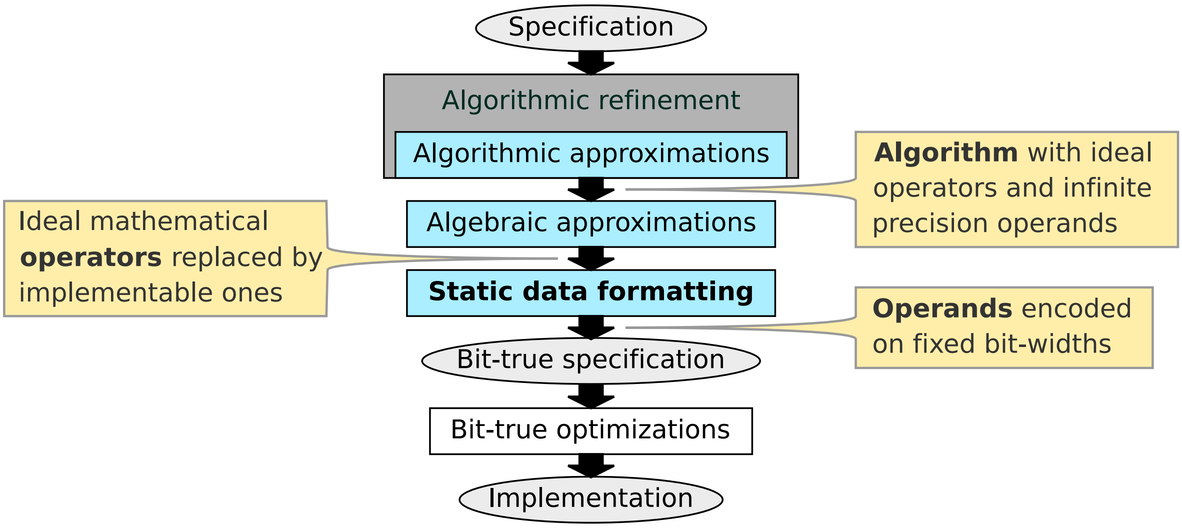
Specification
Algorithmic refinement
Algorithmic approximations
Algebraic approximations
Static data formatting
Bit-true specification
Bit-true optimizations
Implementation
Algorithm with ideal operators and infinite precision operands
Ideal mathematical **operators** replaced by implementable ones
Operands encoded on fixed bit-widths

Figure 1: Design flow proposed to enable approximations in Digital Signal Processing implementations.

When carefully designed, digital signal processing implementations can generally tolerate the approximations imposed by short bit-width operators without a practical degradation of their algorithmic performance (e.g., filter stop band attenuation or cutoff frequency). Accordingly, Figure 1 presents a design flow conceived to minimise implementation cost by exploiting the inherent robustness to approximations of digital signal processing applications. We particularly detail the design steps involved in the refining of the original *ideal precision specification* into a *bit-true specification*. Firstly, a skillful designer will reduce the algorithmic complexity in the *algorithmic refinement* step. For example, the Discrete Fourier Transform (DFT) matrix can be factorised into products of sparse factors (i.e., Fast Fourier Transform), which reduces the complexity from $O(n^2)$ to $O(n \log n)$. Additionally, the algorithmic refinement step can make use of approximations to further reduce the algorithmic complexity. For example, the Maximum Likelihood (ML) detector can be approximated by a near-ML detector [1]. Once the algorithm structure is determined, the subsequent *algebraic transformation* and *static data formatting* steps are responsible of refining operators and signals, respectively. An algebraic approximation can for instance reduce a reciprocal square root operator to a scaled linear function [1]. Finally, the static data formatting step is responsible of encoding every signal in a fixed bit-width. The latter step correspond to the last approximation and results in a bit-true specification, from which all succeeding optimisation transformations (e.g., loop transformations, resource binding, scheduling, etc.) are bounded to be exact.

The static data formatting step of Figure 1 is equivalent to the so-called word-length optimisation process and is the main subject of this chapter. We particularly focus on fixed-point arithmetic (instead of floating point arithmetic) as it generally leads to the most efficient DSP implementations. Yet, fixed-point arithmetic puts all the burden of restraining finite word-length (or quantisation) effects on the designer. Accordingly, in the reminder of this chapter we describe in detail the different stages comprising the fixed-point conversion process (i.e., dynamic range evaluation, word-length optimisation, and accuracy evaluation) and we expose the different techniques available for each stage.

2 Fixed-point arithmetic

2.1 Fixed-point representation

Fixed-point representation is a way to encode real numbers with a virtual binary-point (BP) located between two bit locations as shown in Figure 2. A fixed-point number is made-up of an integer part (left to the BP) and a fractional part (right to BP). The term m designates the integer part word-length (IWL) and corresponds to the number of bits for the integer part when this term is positive. This IWL includes the sign bit for signed numbers. The term n designates the fractional part word-length (FWL) and corresponds to the number of bits for the fractional part when this term is positive. The fixed-point value x_{fxpt} is computed from the following relation

$$x_{fxpt} = -2^{m-1} + \sum_{i=-n}^{m-2} b_i 2^i \tag{1}$$

Numbers in the dynamic range $[-2^{m-1}, 2^{m-1} - 2^{-n}]$ can be represented in this fixed-point format with a precision of $q = 2^{-n}$. The term q corresponds to the quantisation step and is equal to the weight of the least significant bit b_{-n}. The *Q-format* notation can be used to specify fixed-point numbers. For a fixed-point number having an IWL and FWL equal to m

and n, respectively, the notation $Q_{m.n}$ is used for signed numbers and $uQ_{m.n}$ for unsigned numbers. The total number of bits w is equal to $m + n$. In fixed-point arithmetic, m and n are fixed and lead to an implicit scaling factor equal to 2^{-n} which does not change during the processing. The fixed-point value x_{fxpt} of the data x can be computed from the integer value x_{int} of the data x such as $x_{fxpt} = x_{int}.2^{-n}$

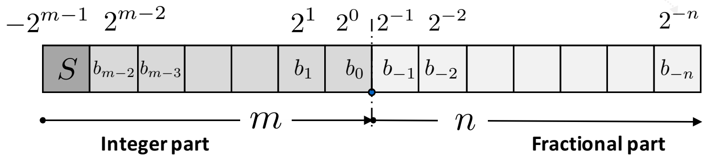

Figure 2: Fixed-point specification

2.2 Format propagation

In this section the rules governing the propagation of the fixed-point formats through operations are described for the different arithmetic operations. Let us consider an operation $\diamond$ having x and y as input operands and z as output operand. Let $Q_{m_x.n_x}$, $Q_{m_y.n_y}$ and $Q_{m_z.n_z}$ be the *Q-format* of the operand x, y and z, respectively.

Addition - subtraction The addition or the subtraction of two fixed-point numbers x and y can lead to an overflow if the operation result is not in the dynamic range of x and y. In this case one more bit must be used to represent the integer part. Thus, dynamic range of the output result must be taken into account. A common IWL, m_c, must be defined to represent the input and the output

$$m_c = \max(m_x, m_y, m_z) \tag{2}$$

where m_z is computed from the dynamic range of the variable z. This IWL allows aligning the binary-point of the two input operands before computing the addition or the subtraction. The fixed-point format of the operation output is as follow

$$\left\{ \begin{array}{rcl} m_z & = & m_c \\ n_z & = & \max(n_x, n_y) \end{array} \right. \tag{3}$$

Multiplication In contrast to the addition or the subtraction, there is no risk of overflow for the multiplication if the format of the output respects the following conditions. Thus, the fixed-point format of the output $z = x \times y$ is obtained from the input x and y fixed-point format with the following expression

$$\left\{ \begin{array}{rcl} m_z & = & m_x + m_y \\ n_z & = & n_x + n_y \end{array} \right. \tag{4}$$

The multiplication leads to an increase of the number of bits to represent the operation output. The total number of bits w_z is equal to $w_x + w_y = m_x + n_x + m_y + n_y$.

Division For the division operation $z = x/y$, the value 0, must be excluded of the divisor y interval $[\underline{y}, \overline{y}]$ leading to the interval $[\underline{y}, -2^{-n_y}] \cup [2^{-n_y}, \overline{y}]$ if we consider the case that $\underline{y}$ is strictly negative and $\overline{y}$ is strictly positive. The IWL of the division output must be able to represent the largest value of the division result. This one is obtained by dividing the largest dividend by the smallest divisor. The largest possible dividend is -2^{m_x-1} while the smallest divisor is 2^{-n_y}.

The FWL of the division output must be able to represent the smallest absolute value of the division result. This one is obtained by dividing the smallest dividend by the largest divisor. The smallest dividend is 2^{-n_x} while the largest divisor is -2^{m_y-1}.

Thus, the fixed-point format of the output $z = x/y$ is obtained from the input x and y fixed-point format with the following expression

$$\left\{ \begin{array}{rcl} m_z & = & m_x + n_y \\ n_z & = & n_x + m_y \end{array} \right. . \tag{5}$$

Like for the multiplication, the total number of bits w_z is equal to $w_x + w_y = m_x + n_x + m_y + n_y$.

2.3 Quantisation process and rounding modes

In DSP applications, a sequence of arithmetic operations leads to an increase of data word-length when multiplication and division operations are involved. To maintain data word-lengths in reasonable range, the number of bits must be reduced. In fixed-point arithmetic, the least significant bits are discarded. Let x', be a fixed-point variable with a word-length of $w_{x'}$ bits. The quantisation process $Q()$ leads to the variable x, depicted in Figure 2, and having a word-length $w = w_{x'} - d$. Let S_x be the set containing all the values which can be represented in the format after quantisation.

Truncation In the case of truncation, the data x is always rounded towards the lower value available in the set S_x:

$$x = \lfloor x \cdot q^{-1} \rfloor \cdot q = \ kq \quad \forall x \in [k \cdot q; (k+1)q[\tag{6}$$

with $\lfloor \cdot \rfloor$, the floor function defined as $\lfloor x \rfloor = \max (n \in \mathbb{Z} | \, n \leq x)$ and $q = 2^{-n}$ the quantisation step. The value x after quantisation is always lower or equal to the value x before quantisation. Thus, the truncation adds a bias on the quantised signal and the output quantisation error will have a non zero mean. Truncation rounding is widely used because of its cheapest implementation. The d LSB of x' are discarded and no supplementary operation is required.

Conventional rounding To improve the precision after the quantisation, the rounding quantisation mode can be used. The latter significantly decreases the bias associated with the truncation. This quantisation mode rounds the value x to the nearest value available in the set S_x:

$$x = \left\lfloor \left(x + \frac{1}{2} q \right) \cdot q^{-1} \right\rfloor \cdot q = \left\{ \begin{array}{ll} kq & \forall x \in [k \cdot q; (k+\frac{1}{2})q[\\ (k+1)q & \forall x \in [(k+\frac{1}{2})q; (k+1)q] \end{array} \right. \tag{7}$$

The midpoint $q_{1/2} = (k + \frac{1}{2})q$ between kq and $(k+1)q$ is always rounded up to the higher value $(k+1)q$. Thus, the distribution of the quantisation error is not exactly symmetrical and a small bias is still present.

The conventional rounding can be directly implemented from (7). The value 2^{-n-1} is added to x' and then the result is truncated on w bits. In the technique presented in [2], the conventional rounding is obtained by the addition of x' and the value $b_{-n-1}.2^{-n}$ and then the result is truncated on w bits. This implementation requires an adder of w bits.

Convergent rounding To reduce the small bias associated with the conventional rounding, the convergent rounding can be used. To obtain a symmetrical quantisation error, the specific value $q_{1/2}$ must be rounded-up to $(k+1)q$ and rounded-down to kq with the same probability. The probabilities that a particular bit is 0 or 1 are assumed to be identical and thus the rounding direction can depend on the bit b_{-n} value.

$$x = \begin{cases} kq & \forall\ x \in [k.q; (k+\frac{1}{2})q[\\ (k+1)q & \forall\ x \in](k+\frac{1}{2})q; (k+1)q] \\ kq & \forall\ x = q_{1/2} \quad \text{and} \quad b_{-n} = 0 \\ (k+1)q & \forall\ x = q_{1/2} \quad \text{and} \quad b_{-n} = 1 \end{cases} \tag{8}$$

The specific value $q_{1/2}$ has to be detected to modify the computation in this case. For this specific value, the addition of the data x with the value 2^{-n-1} has to be done only if the bit b_{-n} is equal to one.

The alternative to this conditional addition is to add the value $b_{-n-1}.2^{-n}$ in every case. Then, for the specific value $q_{1/2}$, the least significant bit b_{-n} of the data x is forced to 0 to obtain an even value. This last operation does not modify the result when b_{-n} is equal to 1 and discard the previous addition operation if b_{-n} is equal to 0. The convergent rounding requires a supplementary addition operation and an operation (DTC) to detect the value 2^{-n-1} and then to force bit b_{-n} to zero.

2.4 Overflow modes

In DSP applications, numerous processing kernels involve summations requiring to accumulate intermediate results. Consequently, the dynamic range of the accumulation variable grows and can exceed the bounds of the values that can be represented leading to overflows. When an overflow occur, if no supplementary hardware is used, the wrap-around overflow mode is considered. For the wrap-around overflow mode, the value x_{wa} of variable x coded with m bits for the IWL is equal to

$$x_{wa} = \left((x + 2^{m-1}) \bmod 2^m\right) - 2^{m-1} \tag{9}$$

with mod the modulo operation. To avoid overflow, the fixed-point conversion process described in the rest of this chapter must be followed conscientiously. Especially, the dynamic range of the different variables must be carefully evaluated for sizing the IWL. For variables having a long tail for its probability density function, the IWL can be large and thus leading to an over-estimation for numerous values. In this case, saturation arithmetic can be used to reduce the IWL. Let consider a variable x coded with m bits for the IWL. In saturation arithmetic, when the value x is lower than -2^{m-1} the value x is set to -2^{m-1}. When the value x is higher than -2^{m-1} the value x is set to $-2^{m-1} - 2^{-n}$.

3 Architecture support for fixed-point

Different targets can be considered to implement DSP systems. These targets provides different trade-offs in terms of flexibility, performance, and energy consumption. Moreover, these targets exhibits different characteristics in terms of word-length granularity supported by the processing resources as depicted in Figure 3. By using custom operators, ASIC and FPGA provide fine-grained word-length operators as detailed in Section 3.1. By using existing hard-wired operators in FPGA (DSP blocks) or in processors only mid-grained and coarse-grained word-length operators are available as detailed in Sections 3.2.

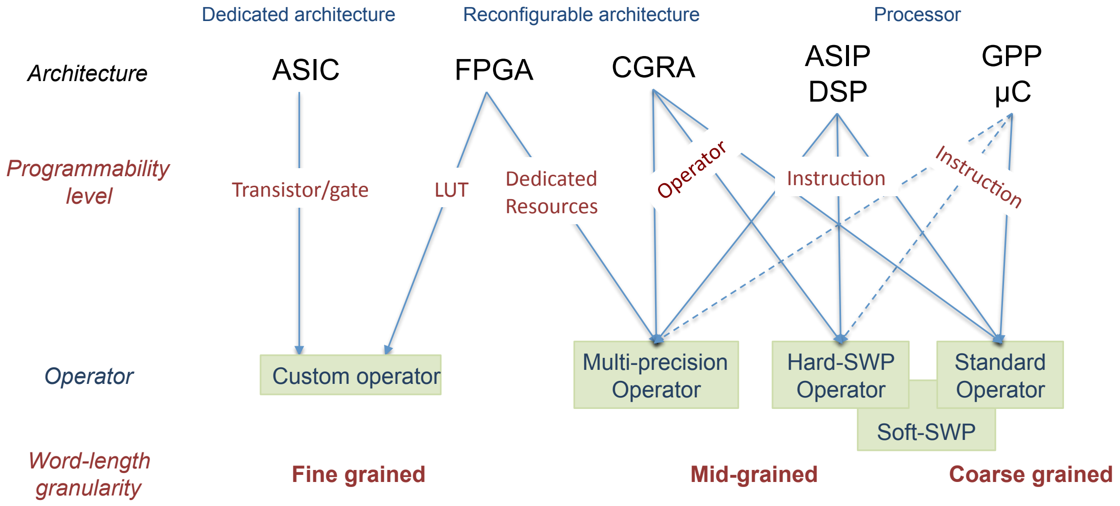

Figure 3: Word-length granularity for the different targets to implement DSP systems.

3.1 Fine-grained word-length operators

Hardware implementation of signal processing applications allows achieving high performance by specialising the architecture to the application. In ASIC, the architecture can be optimised up to the transistor level. In FPGA, the interconnection between specialised arithmetic blocks (DSP), Look-Up Tables (LUT) and registers can be configured to achieve more complex structures. Consequently, in these architectures custom operators can be considered. The operator word-length can be specialised and adjusted bit by bit and thus fine-grained word-length operators can be obtained.

To illustrate the interest of using fine-grained word-length operators, the energy obtained for adders and multipliers using LUTs for a Virtex-5 device is reported in Fig. 4. The energy is given for input word-lengths from 4 to 40-bits for the adder and from 4 to 48 bits for the multiplier. The energy is measured using the technique developed in [3]. Only the dynamic energy consumption (related to the activity in the circuit) is considered. These results clearly show the impact of word-length (wl) on the energy cost metric. For adders, the energy varies linearly from $1pJ$ to $12pJ$ with bit-width from 4 to 40 bits. Multiplier energy varies quadratically according to bit-width when the multiplier is implemented using only LUTs. These results show the interest of adjusting the data word-length bit by bit to minimise the energy consumption.

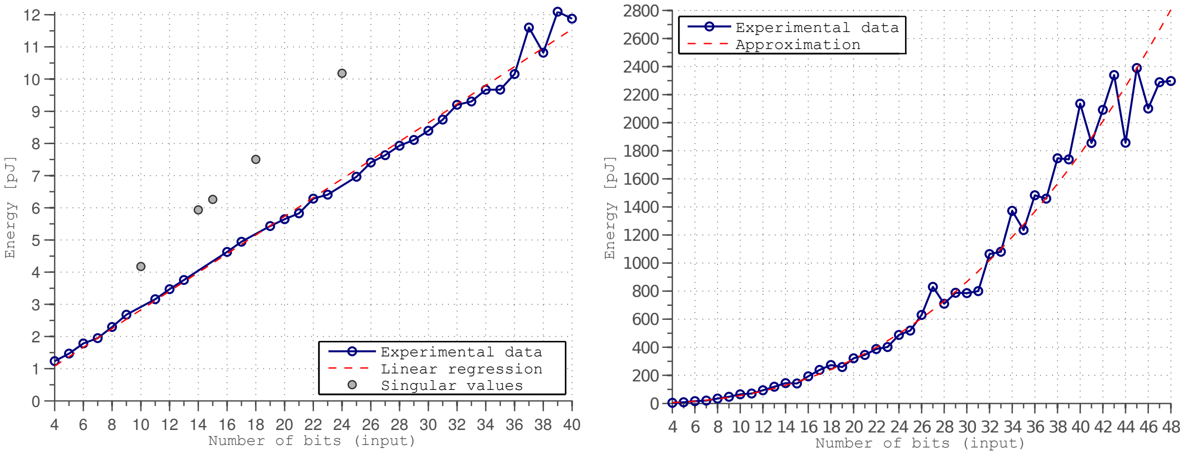

Figure 4: Energy consumption adders (left) and LUT based multipliers (right) with variable *wl* for Virtex-5.

3.2 Mid and coarse-grained word-length operators

Processors based on specialised architecture can lead to efficient implementation of DSP applications. In this context, processor architectures have been proposed [4], [5], [6], [7], which give up the generality of traditional general purpose processors to become far more application specific. Such processors close the gap with ASICs to $2-4\times$, reporting energy efficiency in the order of 1000 MOPS/mW in 40nm CMOS standard cell technology.

This type of processor architectures generally relies on fixed-point arithmetic as it consumes significantly less energy and cycles than floating point arithmetic for most applications. The typical presence of multiple word-length signals in the target applications is clearly beneficial for ASIC implementations, which can reduce area, delay and power consumption by customising the hardware to the application needs. Instead, the benefit that a programmable processor implementation, which includes a fixed bit-width (native word-length) datapath, can achieve from multiple word-length signals seems rather limited and less interesting. The CoolFlux [8] processor from NXP is an example of data-path bit-width customisation. This includes a non-standard 24-bit datapath optimised for the typical bit-widths of audio applications.

3.2.1 Multi-precision operations

To increase the precision compared to the native word-length, multi-precision operations can be considered. The multi-precision operation is decomposed in a set of operations handling native word-length data. In FPGA, for efficient implementation of DSP applications, hardwired multipliers and adders (DSP blocks) are provided. By combining these elements, multi-precision operators can be obtained. Fig. 5 shows the energy obtained for multipliers using DSP blocks for a Virtex-6 device. Using DSP blocks, the energy exhibits a stair effect according to bit-width, which is due to the structure of DSPs constituted of 9-bit elementary operators. When an operation with native word-length operand requires one instruction in a processor, the multi-precision operation requires several instructions. The energy and the latency of multi-precision operations is significantly higher than for operations with native word-length data.

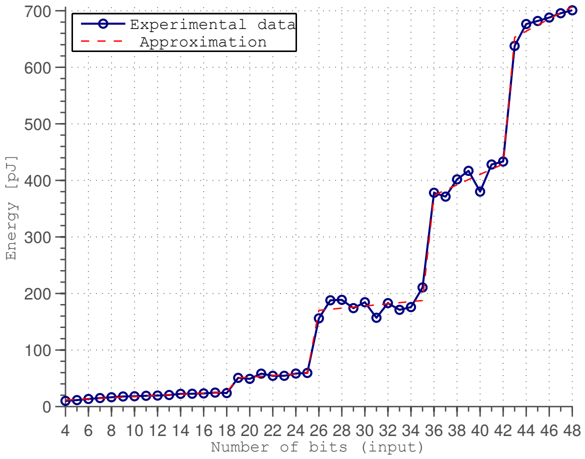

Figure 5: Energy consumption of Virtex-6 multipliers using DSP blocks with variable wl.

3.2.2 Exploitation of sub-word parallelism with SIMD

Alternatively, sub-word parallelism can be exploited to increase processing efficiency at the expense of a reduction of the data word-length compared to the native one. Data-parallel datapaths enable processors to execute a single instruction on multiple independent sets of data (sub-words) in parallel, producing multiple independent results. Considering the deterministic behaviour of the applications under consideration, a software-controlled data memory enables the optimisation of the data layout considering at design time the particular word-lengths.

Thereby, the cost in energy and time associated with the instruction (instruction fetching, operand load, execution and result storage) is shared by all the sub-words. Importantly, the cost per instruction does not increase when operating over a higher number of sub-words [9]. Thus, sub-word parallel processors can for example execute two operations per instructions operating on signals with $N/2$ bits, where N corresponds to the processor's datapath bit-width. The same processor can execute four operations per instructions operating if the signals can be reduced to $N/4$ bits with the corresponding cost reduction per operation. This technique is also known as Single Instruction, Multiple Data (SIMD) and makes execution time and energy consumption dependent on the signal bit-widths. Notice that in order to take advantage of the SIMD the application under execution needs to exhibit enough data-parallelism to fill most of the available sub-words. Hardware SIMD and software SIMD are two different SIMD flavours.

Hard-SIMD Hardware SIMD (Hard-SIMD) is used by many commercial low power and high performance processors. Hard-SIMD requires special hardware support in order to keep the independence across sub-words. For instance, the carry propagation path of a Hard-SIMD adder is either interrupted or propagated in the sub-word boundary depending on a configuration bit. Therefore, every sub-word combination requires a different instruction to generate the corresponding control signals. To keep this overhead low, Hard-SIMD processors typically support a limited number of different sub-words of the same length (e.g. TI C64 DSP supports 1x32-bit, 2x16-bit or 4x8-bit sub-words in a 32-bit datapath [10]). Obviously, the limited number of different sub-words forces the ceiling of the signal to the next available sub-word. This can lead to unused bits, e.g., a 9b signal rounded up to 16b, and therefore a substantial loss in efficiency.

The tendency in processor architectures is to widen datapath bit-width in order to achieve a higher performance. For example the ADRES processor includes a 64b SIMD datapath with support for 4x16-bit operations [11], the NXP EVP includes 256-bit SIMD datapath with support for 16x16-bit operations [12] the AnySP includes a 1024-bit SIMD datapath with support for 64x16b operations[5]. With NEON, a 128-bit SIMD co-processor can be integrated in Arm architecture to provide up to 16 operations on 8b data. Furthermore, Arm has recently introduced the Scalable Vector Extension (SVE) in its 64-bit Armv8-A architectures enabling implementation choices for vector lengths that scale from 128 to 2048 bits.

Word-length optimisation techniques to drive data-type selection for Hard-SIMD processor architecture have been published. A reduction in cycle count by factor of 2 to 3.5 compare to single word-length alternative is reported by Menard *et al.* [13] for a set of important kernels in the wireless application domain. Also, reductions up to a factor of 3 in energy consumption are reported by Novo *et al.* [14] in the execution of the digital baseband processing of a WLAN receiver.

Soft-SIMD Software SIMD (Soft-SIMD) is another flavor of data-parallel processing which does not require operators with specific hardware support to keep sub-word independence [15, 16]. Instead, maintaining sub-words bounds is now the responsibility of the designer, e.g. by inserting guards bits between sub-words to avoid bit rippling across them. Such an approach enables high flexibility in the vector configuration, enabling sub-words of arbitrary length to be operated together as a single word (e.g. a 32-bit datapath can now include 1x12-bit and 2x10-bit sub-words). Hence, the Soft-SIMD can better exploit fine-grained signal word-length information (theoretically down to the bit level) with more versatile sub-word combinations.

However, standard Soft-SIMD suffers from two major drawbacks. On the first hand, only positive numbers can be operated. Guard bits need to be equal for all the sub-words, but due to the nature of 2's complement representation, 0's are required as guard bits for positive numbers and 1's for negative ones. This can be solved by adding an offset to the computed data. This offset needs to be revised at different parts of the processing in order to maintain positive values while optimally utilising the range allocated in the word-length. On the other hand, multiplications require a prohibitive amount of guard bits. The need for many guard bits is related to the word-length growth experienced in a multiplication and has been the main showstopper of such a technique. Fortunately, this is not the case for constant multiplications which can be decomposes in sequences of Shift-Add operations. An extensive description of the Soft-SIMD technique is presented by Novo *et al.* [4]. The latter work implements a video surveillance application on a programmable processor enhanced with Soft-SIMD achieving a very high energy efficiency, which is estimated to be only 2× lower than that of an equivalent ASIC implementation.

4 Fixed-Point Conversion Process

The fixed-point conversion process aims at selecting the set of word-lengths leading to the cheapest implementation while bounding the accuracy degradation to a level that is tolerable

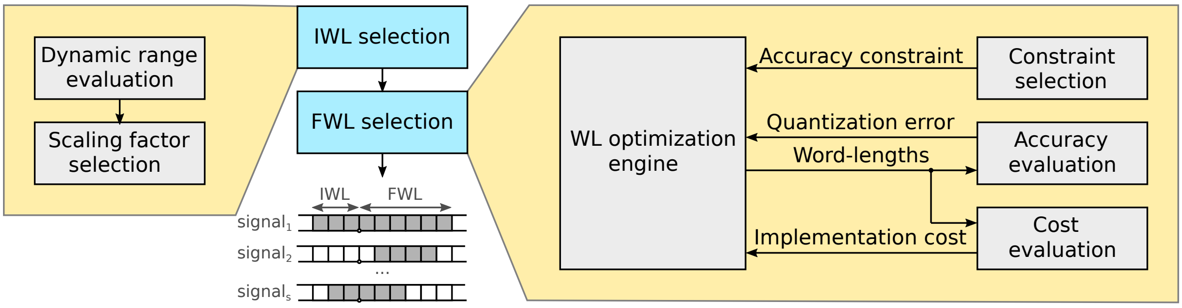

Figure 6: Basic components of the proposed fixed-point conversion process.

by the application in hand. The latter can formally be defined as the following optimisation problem:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimise}} \quad & C(\mathbf{w}) \\ \text{subject to} \quad & \epsilon(\mathbf{w}) \leq \epsilon_{max}, \end{aligned} \tag{10}$$

where $\mathbf{w}$ is a vector containing the word-lengths of every signal, $C(\cdot)$ is an implementation cost function that propagates variations in word-lengths to design objectives such as area or energy consumption, $\epsilon(\cdot)$ computes the degradation in precision caused by a particular $\mathbf{w}$, and ϵ_{max} represents the maximum precision loss tolerable by the application.

From a methodological perspective, the word-length optimisation process can be approached in two consecutive steps: (1) *Integer Word-Length (IWL) selection* and (2) *Fractional Word-Length (FWL) selection*. The IWL selection step defines the left hand limit – or *Most-Significant Bit (MSB)* – and the position of the binary point. The subsequent FWL selection step fixes the right hand limit – or *Least-Significant Bit (LSB)* – of each word-length. Figure 6 shows the basic components of the fixed-point conversion process. Typically, the IWL selection step is designed to avoid overflow errors altogether and consists of the following two subsequent steps:

- The *dynamic range evaluation* step analyses each signal with the objective of setting an upper bound on its magnitude growth.

- The *scaling factor selection* step inserts scaling operations to obtain a valid fixed-point specification. For instance, scaling operations are inserted to align binary point for addition of two fixed-point numbers.

The FWL selection step is usually the main responsible for accuracy loss and can be further divided into the following four interacting components:

- The *World-Length (WL) optimisation engine* basically consists of an algorithm that iteratively converges to the best word-length assignment. It has been shown that the constraint space is non-convex in nature and the optimisation problem is NP-hard [17]. Accordingly, existing practical approaches are of a heuristic nature [18, 19, 20].

- A precise *cost evaluation* of each word-length assignment hypothesis leads to impractical optimisation times as the aforementioned heuristic optimisation algorithms involve many cost and accuracy evaluations. Instead, a practical word-length optimisation process uses fast abstract cost models, such as the hardware cost library introduced by Sung *et al.* [21] or the fast models proposed by Clarke *et al.* [22] to estimate the power consumed in arithmetic components and routing wires.

- The *accuracy constraint selection* block is responsible of reducing the abstract sentence "the maximum precision loss tolerable by the application" into a metric that can be measured by the accuracy evaluation block. Practical examples have been proposed for audio [23] or wireless applications [1].

- The *accuracy evaluation* block evaluates the accuracy degradation metric ϵ for a given word-length $\mathbf{w}$. Existing approaches for *accuracy evaluation* can be divided into simulation-based and analytical methods. Simulation-based methods are suitable for any type of application but are generally very slow. Alternatively, analytical accuracy evaluation methods can be significantly faster but often restrict the domain of application (e.g., only linear time-invariant systems [18]). There are also hybrid methods [24] that aim at combining the benefits of each method.

The remaining of the chapter will cover in greater detail the IWL selection step in Section 5 and the FWL selection step in Section 6. The latter particularly focusses on the WL optimisation engine (Section 6.1) and accuracy evaluation step (Section 6.2).

5 Integer-Part Word-Length Selection

The first stage of the fixed-point conversion process aims at determining the number of bits for the integer part of each signal in the considered system. The goal is to minimise the number of bits while protecting against overflows which degrade significantly the application quality. Firstly, the dynamic range of each signal is evaluated. The different types of techniques available to estimate the dynamic range are presented in Section 5.1. Secondly, the IWL is determined from the dynamic range and the fixed-point format propagation rules. Scaling operations are inserted to adapt fixed-point formats. This process is described in Section 5.2.

5.1 Dynamic range evaluation

The determination of the number of bits for the integer part requires to evaluate the signal dynamic range. Figure 7 presents a classification of the existing techniques used to evaluate the dynamic range.

Critical systems do not tolerate high computational errors. Any overflow occurrence may lead to a system failure or a serious quality degradation. For example, in 1996, the first launch of the Ariane 5 rocket ended in explosion due to software failure. This failure was caused by the overflow of the variable representing the rocket acceleration. Thus, for critical systems, the integer part word-length has to cover the entire range of possible values. In this case, the bounds should be determined by techniques that guarantee the absence of overflow occurrence and allowing to certify the signal dynamic range. Techniques based on interval arithmetic or affine arithmetic satisfy this constraints, but at the expense of an overestimation of the bounds. These techniques are presented in Section 5.1.1. Statistical approaches that determine bounds from a set of simulation results can reduce the overestimation, but can not ensure the absence of overflow occurrence. These techniques are presented in Section 5.1.2.

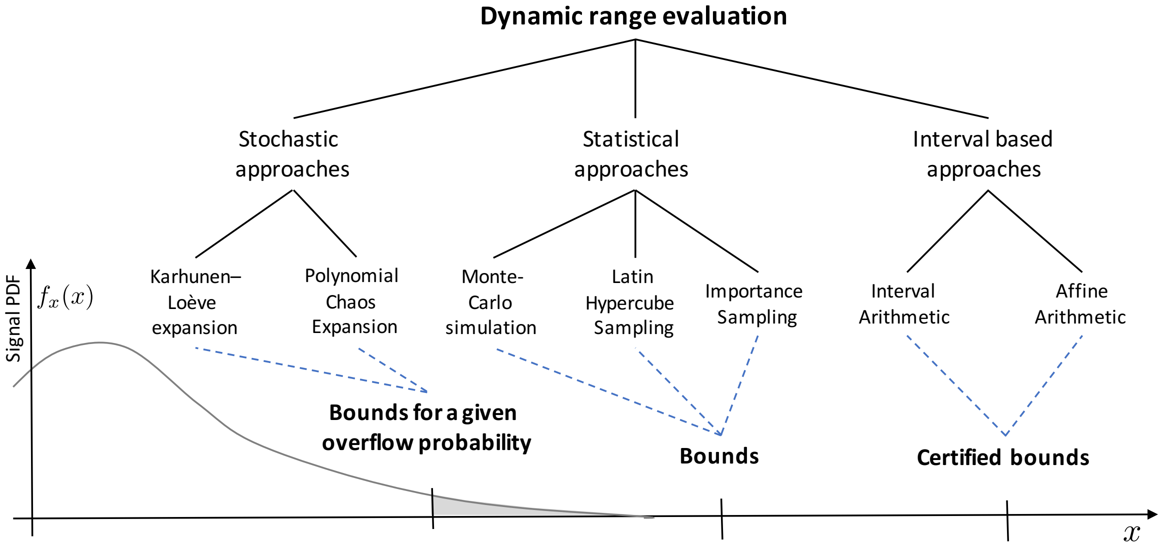

Figure 7: Classification of existing techniques to evaluate dynamic range.

Overflows occur when the number of bits of the integer part is not sufficient. Overflow occurrence degrades the result quality at the system output. However, the hardware implementation cost is unnecessarily increased if the number of bits exceeds the needs. Many systems 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 output system quality degradation. This is translated into an optimisation problem where the integer word-length of each variable of the system is reduced while maintaining an overflow probability lower than the accepted probability [25]. The challenge is to estimate the probability density function (PDF) of the data in order to be able to compute the overflow probability. Approaches presented in 5.1.3 can be considered to estimate the PDF. Stochastic approaches model the variable PDF by propagating data PDF model from the inputs to the system output.

5.1.1 Interval-based methods

Different methods handling intervals have been proposed in the literature to provide certified bounds of the dynamic range of the signals. The aim is to determine the interval of the output by propagating the input interval towards the output. Propagation rules are defined for each type of operation.

IA-based techniques Interval Arithmetic (IA) [26] was probably one of the initial techniques to provide bounds on the uncertainties of complex systems. In IA, a variable x is replaced by its interval $I_x = [\underline{x}, \overline{x}]$. Let us consider operation $\diamond$ having x and y as input operands and z as output operand. The interval of the two input operands x and y are respectively $[\underline{x}, \overline{x}]$ and $[\underline{y}, \overline{y}]$. The interval $[\underline{z}, \overline{z}]$ of the output z is obtained with the following expressions for arithmetic operations

$$\begin{array}{ll} z = x + y & [\underline{z}, \overline{z}] = [\underline{x} + \underline{y},\ \overline{x} + \overline{y}] \\ z = x - y & [\underline{z}, \overline{z}] = [\underline{x} - \overline{y},\ \overline{x} - \underline{y}] \\ z = x * y & [\underline{z}, \overline{z}] = [\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{xy}),\ \max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{xy})] \\ z = x/y & [\underline{z}, \overline{z}] = [\underline{x}, \overline{x}].[1/\overline{y},\ 1/\underline{y}] \quad \text{with} \quad 0 \notin [\underline{y}, \overline{y}] \end{array} \tag{11}$$

The main advantage of IA is that it is able to easily provide certified bounds of all the possible results of a given function. On the other hand, it suffers from three different types of problems [27]: the dependency problem, the cancellation problem, and the wrapping effect. The dependency problem expresses that IA computations overestimate the output range of a given function whenever it depends on one or more of its variables through two or more different paths. The cancellation problem occurs when the width of the intervals is not canceled in the inverse functions. In particular, this situation occurs in the subtraction operations (i.e., according to the IA computation rules, $I_x - I_x = 0$ for a given non-empty interval I_x), what can be seen as a particular case of the dependency problem, but its effect is clearly identified. The wrapping effect occurs because the intervals are not able to accurately represent regions of space whose boundaries are not parallel to the coordinate axes.

One of the earliest works that suggested the application of interval-based techniques to perform finite WL analysis of DSP systems was the finite WL evaluation tool Fridge [28, 29]. This tool performed the so-called worst case estimation, making use of designer annotations to compute the maximum and minimum bounds of the values of the signals. In the computations, they perform the operations using Interval Arithmetic (IA) without specifically referring to it.

Several C++ libraries for IA analysis have been proposed like Profil/BIAS [30], MPFI [31] and BOOST Interval Arithmetic Library [32]. The use of IA has been explicitly suggested in [33] to provide bounds on the allowable range of values of the resources of the datapath of a given circuit. However, in most cases IA provides oversised bounds [34, 35, 36]. As a result, it only provides useful results in systems without feedback loops, or in combination with techniques to reduce overestimation [28, 37].

One of the most intuitive ways to reduce the overestimation provoked by IA is to split the input intervals in sets of smaller sub-intervals covering the same ranges. In the literature, this method has been called Multi-Interval Arithmetic (MIA) [38, 39, 40]. Formally, the ranges of the input signals are represented as unions of P intervals (MIA$_P$), allowing automatic monitoring of the growth of the integer and fractional parts of the variables. MIA was initially suggested to perform fast evaluation of the signal ranges of DSP systems in [38], but it has also been shown that it only provides useful results in acyclic data-flow graphs [34]. The mathematical justification of some of its properties can be found in [39], and the analysis of MIA as a means to perform fast computation of the bounds of the finite WL effects has been given in [34, 38, 40]

AA-based computations Affine Arithmetic (AA) and its extensions have emerged as a popular technique to efficiently compute the bounds of the uncertainties in many types of applications. AA is able to accurately model and propagate several uncertainties through linear systems using simple computation rules. Modified AA (MAA) performs a more accurate propagation of the uncertainties in non-linear systems, at the expense of an exponential increase in the number of elements that need to be computed. Furthermore, the basic elements in MAA do not form an orthonormal basis, so MAA needs to be combined with Polynomial

Chaos Expansion (PCE) techniques to provide accurate and precise results.

Affine Arithmetic In AA [41, 42, 43], each element or affine form consists of a central value plus a series of noise terms. Each noise term is composed of an uncertainty source identifier (the noise symbol) and a constant coefficient associated to it. The mathematical expression is:

$$f_{AA}(\varepsilon) = \hat{x} = x_c + x_0\varepsilon_0 + x_1\varepsilon_1 + x_2\varepsilon_2 + ... + x_n\varepsilon_n \tag{12}$$

where $\hat{x}$ represents the affine form, x_c is the central point, and ε_i and x_i are the i-th noise symbol and its associated coefficient. Operations are classified in affine and non-affine operations. Affine operations (addition and constant multiplication) are computed without error, but non-affine operations need to include at least one additional noise term to provide the bounds of the results. The main advantage of AA is that it keeps track of the noise symbols and cancels all the first-order uncertainties. In non-linear systems AA obtains quadratic convergence, but the increment of the number of noise terms in nonlinear operations makes the computations less accurate and time-consuming. A detailed analysis of the implementation of AA and a description of the most relevant computation algorithms is given in [43].

In the DSP literature, AA was originally proposed to provide bounds of the mantissas of the floating-point descriptions [44]. They also computed the ranges of the signals in fixed-point realisations but only reported accurate results in LTI systems without feedback loops [34, 36]. AA has been commonly used to provide bounds on linear and non-linear systems without feedback loops (linear transforms [45, 46, 47], polynomial evaluations [45, 47], B-splines [45], etc.). However, the bounds provided by AA are only equal to the exact ranges of the results in linear sequences of operations, and some oversizing is introduced otherwise [43, 34]. When this oversizing is not accumulated in the loops of the realisation under study, the difference between the signal ranges and the bounds of the AA-based simulation is small. However, if the oversizing is accumulated in the loops, the bounds of the AA-based simulation do not provide useful results [48]. This characteristic is particularly important in systems where the amount of feedback of the loops is large (for example, in quantised IIR filters whose poles are close to the unit circle [48, 46]).

Modified AA Modified AA (MAA) [49, 50] is an extension of AA that includes the terms of order higher than one in the affine forms. In this technique the affine forms are polynomials in which the independent variables represent uncertainties in the reference interval $[-1, 1]$. Its mathematical expression is:

$$f_{MAA}(\hat{\varepsilon_x}) = \sum_{i=0}^{m} d_i \varepsilon_x^i \tag{13}$$

This expression is an extension of AA in which all the high order terms are taken into account. In the original works, MAA is used for polynomial evaluation and algebraic curve plotting in 2D [49, 50]. However, since the monomials of MAA do not form an orthonormal basis, this approach can lead to inaccuracies in the propagation of the affine forms [51].

Combined application of MAA and PCE The combined application of MAA and Polynomial Chaos Expansion (PCE) accurately propagates the uncertainties through the non-linear descriptions of the systems. For the MAA-based PCE description, its mathematical expression is as follows:

$$f_{PCE}(\hat{\varepsilon_x}) = \sum_{i=0}^{\infty} \alpha_i \psi_i(\varepsilon_x) \tag{14}$$

where α_i are the coefficients of the expansion, and $\psi_i(\varepsilon_x)$ are the orthonormal polynomials of ε_x [52, 51]. Due to the formulation of the PCE, the optimal selection of the polynomial basis depends on the type of random variables being evaluated [52]. The Hermite PCE is best suited to evaluate gaussian random variables, which is the most common type of uncertainty in the input signals [53], and the Legendre PCEs is best suited to propagate uniform random variables [54]. The disadvantage of this approach is that number of polynomials required to form the basis of the PCE significantly increases with the dimension and order of the expansion. For this reason, in practice the number of terms of the expansion is limited to small numbers (e.g., order 3 and dimension 4).

Combined application of MAA and ME-gPC The Multi-Element generalized Polynomial Chaos (ME-gPC) technique is aimed at providing more accurate results than PCE, particularly in systems with discontinuities, at the expense of additional computations. ME-gPC proposes the decomposition of the N-dimensional random input space $B = [-1, 1]^n$ in a set of non-overlapping sub-spaces $B_k = \prod_{i=1}^{n}[a_i^k, b_i^k)$. In order to maintain the orthogonality properties of Legendre Chaos, the new random variables ψ_i^k of each sub-space must be re-scaled to $[-1, 1]^n$. In ME-gPC, the coefficients of the expansion, α_i^k, are obtained by solving a linear system in each sub-space. This is done by choosing $m+1$ uniform grid points in $[-1, 1]^d$ and comparing them with their corresponding values in the original space. The mathematical expression of the MAA-based ME-gPC is as follows:

$$f_{ME-gPC}(\hat{\varepsilon}_x^k) = \sum_{i=0}^{\infty} \alpha_i^k \psi_i(\varepsilon_x^k) \tag{15}$$

With the coefficients obtained using this method, PCE can be locally applied to the different elements. Later, the global statistical moments can be reconstructed by combining all the partial results. While the partitioning described here can be statically performed before the analysis begins, the authors in [55] propose a dynamic adaptive criterion to do it more efficiently. Instead of splitting the domains of all the random variables in the sub-space, only the most sensitive ones are selected for the partition. Once the spaces and random variables to split have been chosen, the partitioning is performed and the process is repeated until no new elements are generated.

5.1.2 Statistical methods

Statistical methods for dynamic range evaluation are based on performing a number of simulations that compute the relevant statistics of the signals of interest. The number of simulations is not as large as in the evaluation of the fractional part, but the techniques and tools indicated in Section 1.6.2.1 can also be applied here, or in combination with the techniques presented below. In this case the simulations are applied over the unquantised description of the realisation. The objective is to obtain the statistics of the range of each signal, and then the number of bits required to represent the integer values in each case (integer WLs).

Among the techniques based on probabilistic simulation, the most relevant are: (1) the Monte-Carlo (MC) method, (2) Latin Hypercube Sampling (LHS), (3) Importance Sampling (IS) and its variants, and (4) Statistical Intervals. Each of them is briefly described below.

The Monte-Carlo method The MC method is based on randomly generating a set of test vectors according to the PDFs of the inputs, processing them through the system realisation, and storing the results for further analysis [56, 57, 58, 59, 60, 37, 61]. This analysis includes the computation of the statistical parameters of the relevant signals (mean, variance, higher order statistics, and even the PDF in some cases [60, 62]), and the selection of the required integer WLs of each signal. The MC method has the advantage of being safe and conceptually simple, which has contributed to become one of the most popular methods for dynamic range evaluation. However, this method requires a very large number of simulations and tends to be very slow, especially when high-precision results are required or when the number of input signals is high.

One of the implicit theorems in the MC method is the Central Limit Theorem, which states that the sum of N independent and equally distributed variables tends to generate a normal distribution when N tends to infinity [59]. On the other hand, the confidence interval is defined as the range of values whose probability of containing the mean is above a certain threshold [63, 64]. The approximation

$$n \approx \frac{10}{P_e} \tag{16}$$

provides the number of samples n required to ensure that the estimated value μ is contained within the interval $[0.5, 2] \cdot \mu$ with an error probability P_e of 97%. Consequently, this value provides the minimum number of simulations needed to obtain reliable results with a predefined error probability [57, 59].

The main advantage of this approach is that all types of realisations are supported. The simulations can be done with system-level design tools such as SPW [65], SystemStudio [66], Matlab-Simulink [67], Ptolemy [68], etc.

Latin Hypercube Sampling LHS is a variant of the MC method based on dividing the PDFs of the input signals into equiprobable parts, and combining the samples associated to each part according to their respective correlations [69, 70]. The selection of samples within each part is done randomly, as in the MC method. There are works that evaluate the behavior of this method with respect to MC. However, they conclude that the accuracy of the results is almost similar (for a given number of samples), but the processing speed is approximately 12 times lower [59].

Importance Sampling. Extensions and variants The IS method consists on modifying the PDFs of the input signals by means of weight functions to enhance the parts of the probability function that have greater impact on the results. These transformations significantly reduce the variance of the estimator to be computed [57, 71, 72]. The new PDF is generated by scaling or moving the original PDF in a controlled manner, and generating the results through MC simulations. Afterwards, the results are corrected according to the displacements made from the initial transformation. This method achieves considerable reductions in the variance of the estimator (up to 10^3% or 10^8%, depending on the application). This means that the number of simulations needed to obtain the reliability of the MC method is significantly reduced. The main drawback of this technique is that, assuming certain predefined initial conditions are met, the precision is very dependent on the system under analysis (especially on the amount of memory and the non-linearities). In addition, the selection of the regions to be displaced and how to make these movements is a very complex issue that ”requires further study” [71].

The most relevant extensions of IS for systems with memory are regenerative simulation, and simulation based on events and conditioning [58, 73]. The main variants to improve the accuracy of the simulations are: Large Deviation Theory (LDT) [58, 74, 75], Selection of Sub-optimal Distributions [76, 77, 74, 78], Adaptive IS [78], Extreme-Value Theory [79, 80] and Quasi-Analytical Theory [57].

Statistical Interval-based Computations As detailed in the Section 1.5.1.1, Interval Computations have been widely used in evaluating the dynamic range of the signals. Due to their improved computional speed, some authors have proposed the incorporation of statistical values to the intervals [81, 82, 83, 52]. Using this representation, interval $[a, b]$ represents a uniform distribution in the range of that interval with an associated probability $p = 1/(b-a)$.

Statistical Interval-based Computations is divided in three tasks: representation of the input statistics using intervals, propagation of the interval parameters, and computation of the output statistics. To provide good results, each of them must be accurately performed, avoiding the introduction of oversizing in any of them. A detailed description of these tasks, with the expressions to perform the required computations, can be found in [83]

Since Interval Arithmetic provides conservative bounds of the results, the PDFs computed using Statistical IA are not accurate: The probabilities of the tails of the distributions are greater than the reference ones, and the central values have smaller probabilities than their respective counterparts. Instead, extensions of IA provide much more accurate results. In this sense, if all the interval computations are accurately performed (i.e., without any oversizing), the computed statistical parameters correspond to the exact results. As a rule of thumb, the more accurately the computations are performed, the more accurate PDFs are provided.

Affine Arithmetic has shown to provide exact results in LTI systems [83, 52], and good first-order approximations for quasi-LTI systems without feedback loops [82]. In the general case, the combination of Modified AA with PCE or ME-gPC techniques [52, 84] provides the most accurate results.

5.1.3 Stochastic methods

Although the most common practice today for dynamic range estimation is still profiling (also referred as simulation), this approach requires exceeding long computation times, and in many cases limited confidence on the accuracy is obtained.

Stochastic methods (also referred as analytical) have emerged and aim at reducing these two limitations. While many advances have been made here, most existing work either makes an ”inadequate treatment” of the signal correlations [53], or requires very long computation times.

In the general case, accurately managing abrupt non-linearities requires capturing the high order correlations of the signals, which in turn demands processing a very large number of parameters. Hence, the adaptation of the evaluation method to the system characteristics is essential to perform efficient computations.

While the use of impulse-response evaluation techniques for LTI system or quasi-LTI systems is the most efficient option [85, 54, 86], the best approach for the evaluation of the dynamic ranges in non-linear systems is the application of PCE and ME-gPC techniques. PCE captures the non-linearities in polynomial systems [53, 52, 51], and ME-gPC provides better results with discontinuities are present in the realisation [51, 84].

The most recent approaches on this area select the most relevant parameters (i.e., they keep track of these elements up to a given number of terms) to obtain the results as fast as possible in the minimum computation time [87, 84].

5.2 IWL determination and insertion of scaling operations

The IWL is determined by propagating the IWL thought the operations from the inputs to the outputs with the help of the dynamic range evaluation results. This propagation process uses the format propagation rules provided in Section 2.2.

To illustrate this stage, let us consider the sequence of operations depicted in Figure 8. The data d is the output of the operation $\mathcal{O}_j$ and the input of operation $\mathcal{O}_k$. Let m_s, m_d and m_i be respectively the IWL for the operation $\mathcal{O}_j$ output, the data d and the operation $\mathcal{O}_k$ input.

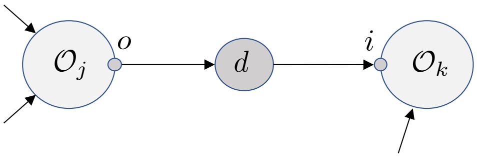

Figure 8: Example of a sequence of operations: data d is the output of the operation $\mathcal{O}_j$ and the input of operation $\mathcal{O}_k$.

The IWL of the signed data d is computed from its dynamic range $[\underline{x}, \overline{x}]$ with the following expression

$$m_x = \max\left(\lfloor \log_2(|\ \overline{x}\ |) \rfloor + 2, \lceil \log_2(|\ \underline{x}\ |) \rceil + 1\right). \tag{17}$$

The IWL m_o is computed from the propagation of the IWL of the operation $\mathcal{O}_j$ inputs with the help of the rules provided in Section 2.2. A scaling operation is inserted at the operation $\mathcal{O}_j$ output if $s_o = m_d - m_{o_j}$ is strictly negative. In this case, a left shift of s_o bits is required to modify the IWL. It means that the IWL of the operation $\mathcal{O}_j$ was too important compared to the data dynamic range of d and the s_o most significant bits of x are a copy of the sign bit and can be discarded.

For multiplication and division, the IWL m_i is equal to m_d and no scaling operation is required at the operation $\mathcal{O}_j$ input. For addition and subtraction, a common IWL m_c for the input and the output must be determined and $m_i = m_c$. A scaling operation is inserted at the operation $\mathcal{O}_k$ input if $s_i = m_{o_i} - m_d$ is strictly positive. In this case, a right shift of s_i bits is required to modify the IWL of the data d. It means that supplementary bits are required for the addition or subtraction $\mathcal{O}_j$ to avoid overflow.

6 Fractional-Part Word-Length Determination

The ultimate goal of fixed-point refinement is to enable the implementation of systems based on mathematical computations while bounding the implementation cost to a reasonable figure. Ideally, the designer seeks the minimisation of the cost given a maximum allowed overall degradation. The minimisation of the cost is provided through finding a suitable set of word-lengths that comply with the accuracy constraints while minimising the cost. A description and comparison of the most popular optimisation techniques can be found in 6.1. The techniques used to check the accuracy of the system along the optimisation process are described in 6.2

6.1 Word-length optimisation

As already stated in Section 4, the word-length optimisation process is defined as

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimise}} \quad & C(\mathbf{w}) \\ \text{subject to} \quad & \epsilon(\mathbf{w}) \leq \epsilon_{max}, \end{aligned} \tag{18}$$

The problem is to find $\mathbf{w}$, a vector containing the word-lengths of every signal, that minimises a cost function $C(\mathbf{w})$, complying with a maximum error degradation ϵ_{max}. The function $\epsilon(\mathbf{w})$ computes the quality degradation due to reduced precision caused by a particular $\mathbf{w}$. The problem depicted in Eq. 18 aim at optimising, for each signal, the fractional part word-length (FWL) $\mathbf{n}_i = \mathbf{w}_i - \mathbf{m}_i$ with $\mathbf{m}_i$ the IWL obtained from the process presented in Section 5.

Depending on the thoroughness of the optimisation process, the implementation cost can be considerably reduced. The fast, but far from optimal, uniform word-length approach has been progressively replaced by the cost-effective multiple word-length approach [88, 18, 89, 90].

In this section we present several techniques to perform the fine-grain optimisation required to obtain multiple word-lengths implementations, focusing on finding the FWL for each signal. These classic techniques cannot be applied directly to industrial-size designs having a great number of variables to optimise. Hierarchical approaches [91, 92] have been proposed to decompose the global optimisation problem into several optimisation problems with a reasonable number of variables to optimise.

6.1.1 Optimisation algorithms

This subsection focuses on automatic optimisation procedures that enable to obtain a set of word-lengths that minimises cost, finding the value of the FWL $\mathbf{n}_i$ for each signal. The optimisation methods presented here have been divided into different categories, sorted from higher to smaller computational complexity. A comparison between the different techniques can be found at the end of the section.

Optimal approaches A straight approach to optimisation is that of *complete search* where all the word-length combinations within a wide space are assessed and the solution that minimises cost is selected [21, 61, 93]. The search requires the computation of lower and upper bounds for all word-lengths to constrain the search space. It is important to bear in mind that there is a chance, small though it may be, that the global minimum is not found within these bounds, since the quantisation error at the outputs of a quantised system is non-monotonic and non-linear with respect to the word-lengths [18].

In [18] a Mixed Integer Linear Programming (MILP) of the quantisation problem is formulated for LTI algorithms. MILP requires expressing the optimisation problem by means of equations and inequalities that are fed to a solver. The quality degradation used is as the the variance of the quantisation noise, derived with an analytical accuracy evaluation technique that considers a discrete noise model and signal branching (see Section 6.2). The final formulation leads to an explosion of variables and constraints for medium/large size algorithms with solving times around several days. This work was extended to consider high-level synthesis in [94].

Optimisation in the continuous domain There are also approaches where the optimal techniques are applied to solve the quantisation problem assuming *real* word-lengths. The obtained solution is then refined looking for a feasible realisation with integer word-lengths. Optimality is not guaranteed, but the authors show a reduction in optimisation time.

The idea was first proposed in [95] and three simplifications were performed: i) the word-lengths are real numbers; ii) cost is a linear combination of the word-lengths; and, iii) the quantisation noise of a signal only depends on its final word-length. The optimisation is then based on an approximate solution to the problem using the Lagrangian multipliers that results in a linear expression of the cost that is a function of the word-length and the so-called noise gains. The noise gains relate the word-lengths with error.

In [96] the error estimation is performed by means of interval arithmetic and the worst-case error contribution of each signal to the outputs is computed (as in [88]). Also, the objective function is linear, since they use the summation of word-lengths as an implementation complexity measure. Thus, the optimisation problem can be solved by Linear Programming, reducing the computation time.

In [97] a nonlinear iterative algorithm is presented. This algorithm is applied to find the real word-lengths that minimise area. The area cost is estimated using a quadratic function for multipliers and a linear function for adders. The convergence of the optimisation method is analysed and bounds on the word-lengths are given. The error estimation is based on analytical techniques, using the Signal-to-Quantisation-Noise Ratio (SQNR) to steer the optimisation (see Section 6.2.2).

A quadratic objective function is addressed in [22] by means of sequential quadratic programming. The main additions of this work in comparison to the other continuous-domain approaches are that it considers the discrete noise model from [98] and also the noise correlation in signal forks.

Stochastic optimisation Stochastic optimisation is based in heuristics that refine the word-lengths in a random but controlled fashion. An efficient stochastic algorithm is simulated annealing (SA), which is based in changing the values of the problem variables by means of the so-called movements. Each time a movement is performed, the resulting cost as well as the output error of the current solution are computed. If the noise constraint is met, then, if the cost is smaller than the current minimum one, the movement is accepted, otherwise, it is accepted with a certain probability. This probability decays with time (by means of a decreasing *temperature*). [99] presented a SA procedure where the movements are based in the use of a Markov-chain. Also, in [100] a similar approach is presented and some comparison results with the heuristics in [101, 18] are provided.

Constrained search Constrained search algorithms perform a thorough but not complete design space exploration. In [102], the *exhaustive search* is presented and it starts with the computation of the minimum word-length solution. First, each signal is set to the minimum word-length that complies with the error constraint while the rest has *infinite* precision, and then, all signals are assigned simultaneously their minimum word-lengths. The next phase is to increase word-lengths until the error constraint is met. This is performed by distributing b bits – starting with $b = 1$ – among several signals are tested. If the current value of b does not result in complying with the error constraint, B is increased.There is a chance of getting stuck in a local minimum.

[101] proposes a *branch and bound* technique. The algorithm starts with a suitable uniform solution and it starts decreasing the word-lengths until reaching a minimum word-length bound. First it assesses the error produced by reducing one unit each current word-length individually. The word-length sets that do not comply with the maximum error are discarded, while for the rest, the resulting cost is recorded. Then, the one-unit reduction is applied to each remaining word-length set. Thus, a tree of solutions is created dynamically and the solution that produced the minimum cost is selected. Again, the solution could be a local minimum.

Local search [101] proposes a gradient descent optimisation coined *local search*, that we referred to as *Max-1*$_{cost}$. The algorithm starts with a valid uniform solution, decreasing the word-lengths until reaching a minimum word-length bound. Each word-length is reduced one bit temporarily and the cost is computed. Then, the signal that produced the maximum cost reduction is then reduced permanently and the word-length reduction process starts again. [61] uses error to steer the optimisation instead of cost.

The *Max-1*$_{cost}$ idea is extended in [18] where the word-lengths are reduced individually until the maximum error constraint is violated. The signal that produced the maximum cost reduction is decreased one bit permanently. Being less dependent on local information, the chances of falling in a local minimum are reduced. We referred to this approach as the *Max-1*$^{+}_{cost}$.

In [61, 103], a gradient ascent algorithm (*Min+1*) starts with the minimum word-length solution and then it tests all possibilities of increasing signals 1 bit. After all signals have been covered the signal that produced the maximum error decrement is chosen and its word-length is increased one unit.

The *Min+b* [19] extend the *Min+1* so that a budget of b bits ($b = [1, 2, \ldots b_{max}]$) is distributed among the signal word-lengths.

[104] elaborates on an idea sketched in [105], presenting the *complexity and distortion search*, that we referred to as *Max-1*$_{err+cost}$. The *Max-1* algorithms is applied but the decision of the candidate to be reduced is based on the minimum gradient of $\alpha_c \cdot cost + \alpha_d \cdot error$, with $\alpha_c + \alpha_d = 1$. The authors claim that the optimisation time is reduced in comparison to considering only cost or only error gradients.

The *evolutive* procedure of [61] starts with all signals with floating-point format. Then, a signal is selected and the minimum word-length that complies with the requirements is found. The signal is permanently assigned the minimum word-length plus one unit. Then another signal is chosen and the same procedure is applied. The method stops when all signals are quantised. The method is extended in [19] with an final extra phase that applies *Max-1*$_{cost}$ to the solution obtained.

The *Max-1*$_{all}$ [21] starts with the minimum word-length solution and it increases all signal word-lengths one bit until the specifications are met.

The *preplanned* search of [103] requires an initial phase to compute the sensitivity of each signal word-length. The sensitivity information enables to preplan the search steps, starting with the minimum word-length solution and increasing the signal with higher sensitivity for the current word-length set. The procedure stops when the error is below the permitted value.

The *hybrid* procedure of [19] first applies *Min+1* and it feeds the obtained solution to the *Max-1*$_{err}$. Another method that combines two optimisation techniques is the *heuristic* of [19] that first applies *constrained search* and then *Max-1*$_{cost}$.

The algorithm proposed in [106] is based on Greedy Randomised Adaptive Search Procedure (GRASP) and combines local search and stochastic optimisation. This algorithm is a multi-start process consisting of two phases: construction phase and local search phase. In the construction phase, a randomised search algorithm is used to find a sub-optimal solution and then, from this solution, a local search (tabu search) is applied to refine the solution. The random aspects allow avoiding local minimums.

6.1.2 Comparison of optimisation techniques

It is not an easy task to quantitatively compare the performance of the different techniques proposed, as they usually include different setups and no common benchmark suites have yet been accepted by the research community. Still, in this section we will try to summarise the main conclusions drawn in papers such as [61, 19, 93], etc.

The approaches *Min+b*[19], *Min+1*[101], *evolutive*[61], *hybrid*[19], *heuristic*[19], *simulated annealing*[99], *Max-1*$_{err}$[19], *branch and boud*[101], and *preplanned*[103] are compared in [19]. The average number of bits of all the signals and the number of optimisation iterations are used as comparison metric. All methods are able to reach the optimal solution if the number of operations of the benchmark is small. Also, for many cases, the methods that do not impose the minimum word-length bound obtain better results. As expected, the algorithms requiring longer computation times were *evolutive*, *SA*, *exhaustive* and *branch and bound*. The combination of two optimisation methods such as *hybrid* and *heuristic* produced the best minimisation. The *exhaustive* method does not perform well because of the lower bound limitation. *SA* does not lead to optimum results since the number of iterations was limited (see [100] to see some examples of the actual performance of SA). The approach *preplanned* is the fastest for the benchmarks tested. It must be stressed that the set of benchmarks used is limited, mainly formed by small-size examples, and that using as a metric the average number of bits does not directly relate to cost.

The approaches *complete search* [21, 61, 93], *exhaustive* search [102], *Min+1* [61, 103] (called *sequential search*), *preplanned* search[103], *Max-1*$_{cost}$ [101], *Max-1*$_{err}$ [61], *evolutive* search [61], *Min+1* [103, 61], *branch and bound* [101] and *Max-1*$_{err+cost}$ [104] are tested and compared in [93] for a optimisation with 5 variables. First, the results yield that the *complete* search requires a number of iterations 4 to 5 order of magnitudes bigger than the rest of options. Again, *preplanned* emerges as the fastest approach although the results are not quasi-optimal and they are highly dependent on the input vectors used for simulation. After that, the paper compares the effect of using error or cost to guide the optimisation, applying *Max-1*$_{cost}$, *Max-1*$_{err}$, *Min+1* and *Max-1*$^{cost}_{err}$ to small-size benchmarks. To summarise, the use of cost gradients seems to produces better final costs, the use of error gradients seems to accelerate the optimisation process and, finally, the use of both (*Max-1*$^{cost}_{err}$) leads to optimisation times as reduced as *Max-1*$_{err}$ and always provides always the best costs.

A comparison of *SA*, *Max-1*$_{cost}$ and *Max-1*$^{enh}_{err}$ can be found in [100]. Regarding computation time the fastest algorithm is *Max-1*$_{cost}$. *Max-1*$^{enh}_{cost}$ takes times one order of magnitude bigger, while *SA* times are 2-3 order of magnitude bigger. In terms of cost minimisation, the average reduction of $\textit{Max-1}_{err}^{enh}$ vs. $\textit{Max-1}_{err}$ is around 5%, while there are a few cases where $\textit{Max-1}_{err}$ performs better. Comparing *SA* and $\textit{Max-1}_{err}$, the latter always performs better and the average cost reduction is around 10% for the benchmarks tested.

Finally, it is interesting to mention works where industrial-size design are addressed in practical time. The key to do so is to use hierarchical methods, where the quantisation problem is subdivided in several independent problems. In [91] quantisation goes as follows. First, the systems is divided in top-level entities – which basically are the main processing blocks (i.e. FFT, channel equaliser, etc.). Then, the top-level signals – the inputs and outputs of top-level entities – are quantised complying with the error constraint and following a cost estimation reduction criteria. Finally, each top-level entity is quantised independently given the error constraints imposed by the quantisation of the top-level signals. The method provides a practical solution to the complexity explosion of word-length optimisation. Also, in [92], a mixed analytical and simulation hierarchical approach is presented where significant design time reductions are reported.

6.2 Accuracy evaluation

Fixed-point arithmetic leads to unavoidable error between the finite precision values and the infinite precision ones. Such errors, due to signal quantisation, degrades the quality of the application output. Thus, the data word-lengths can be reduced until the effect due to signal quantisation on the application quality metric are tolerable. The challenge is to propose technique to evaluate efficiently the effect due to signal quantisation. In the word-length optimisation process, the fixed-point accuracy is evaluated numerous times to select the best word-length combination. In the rest of this section, the techniques available to analyse the effect of signal quantisation are described. In LTI systems, the Coefficient Quantisation (CQ) has an impact on the system frequency response. The CQ process performs a measurement of the degradation of the system response with respect to the quantisation of the coefficients of the realisation. In the analysis of the quantisation effects, these parameters are commonly the first ones to be determined, since they indicate the most convenient structure to perform the system operation.

The CQ has traditionally been evaluated using the so-called Coefficient Sensitivity (CS) [107, 108]. Although this parameter was originally defined for LTI systems, since its operation is fully described by $H(z)$, its current use has been extended to non-linear systems.

6.2.1 Simulation-based approaches

The quantisation error can be obtained by extracting the difference between the outputs of simulation when the system has a very large precision and when there is quantisation (bit-true fixed-point simulation). Double-precision floating-point simulation is usually considered to be the reference given that the associated error is definitely much smaller than the error associated to fixed-point computation. Different error metrics can be computed from the quantisation error obtained from this simulation. The main advantage of simulation-based approaches is that every kind of application can be supported. Fixed-point simulation can be performed using tools such as [65, 66, 67, 68].

Different `C++` classes to emulate the fixed-point mechanisms have been proposed, such as `sc_fixed` (*SystemC*) [109], or `gFix` [110]. Providers of high-level synthesis tools have developed their own fixed-point data-types : `ac_fixed` (*Algorithm C Data Types*) [111] and `ap_fixed` (*Xilinx Data Types*). Indeed, for hardware design, the availability of fixed-point data-types is crucial to specialise the data-path with arbitrary word-length and to provide bit-accurate simulation. The `C++` class attributes define the fixed-point parameters associated to the data. Bit-true operations are performed by overloading the different arithmetic operators. During the execution of a fixed-point operation, the data range is analysed and the overflow mode is applied if required. Then, the data is cast with the appropriate quantisation mode. Thus, for a single fixed-point operation, several processing steps are required to obtain a bit true simulation.

To illustrate emulation of fixed-point mechanisms, the *SystemC* fixed-point data types are considered. The parameters associated with each fixed-point data types are the data word-length (WL), the integer part word-length (IWL), the quantisation mode (QM), the overflow mode (OM) and the number of saturation bits (N). Seven quantisation modes are provided: two for the truncation and five for the rounding. Four overflow mode are provided: one for the wrap-around and three for the saturation. Signed and unsigned data types are available. Normal and fast data types are available. By limiting the precision to 53-bits for the fast data-types, the execution time of fixed-point operation can be significantly reduced. Static or dynamic attribute initialisation are available. The `fixed` data types are template classes for which the parameters are set via variable declaration. The parameter values are static and known at compile-time allowing software optimisation to reduce execution time. The `fix` data types are regular C++ classes for which the parameters are set via their constructors or via parameters context. The parameters are dynamic and can be modified at run time. These three types of alternative lead to eight different data types: `sc_fixed`, `sc_ufixed`, `sc_fixed_fast`, `sc_ufixed_fast`, `sc_fix`, `sc_ufix`, `sc_fix_fast`, `sc_ufix_fast`.

However, these techniques suffer from a major drawback which is the long simulation time [112]. This becomes a limitation when these methods are used in the data word-length optimisation process where multiple simulations are needed. The simulations are made on floating-point machines and the extra-code used to emulate fixed-point mechanisms increases the execution time between one to two orders of magnitude compared to traditional simulations with native floating-point data types [29, 113]. Besides, to obtain an accurate estimation of the statistical parameters of the quantisation error, a great number of samples must be taken for the simulation. This large number of samples combined with the fixed-point mechanism emulation lead to very long simulation time.

6.2.2 Analytical approaches

Metrics to evaluate quantisation error accuracy In the case of analytical approaches, a mathematical expression of a metric is determined. Determining an expression of the application quality metric for every kind of application is generally an issue. Thus, the application quality degradations are not analysed directly in the fixed-point conversion process and an intermediate metric which measures the fixed-point accuracy must be used. Different metrics can be used. This accuracy can be evaluated through the bounds of the quantisation errors [114, 115], the number of significant bits [116], or the power of the quantisation noise [117, 118, 119]. The shape of the power spectral density (PSD) of the quantisation noise is used as metric in [120] or in [121] for the case of digital filters.

For the metric that computes the quantisation error bounds, the maximum deviation between the exact value and the finite precision value is determined. This metric is used for critical systems when it is necessary to numerically validate the final quality and to ensure that the error will not surpass a maximum deviation. In this case, the maximal error is determined in the worst-case. Similar techniques as those used to determine data dynamic range can be used to determine the quantisation error bounds.

For signal processing applications, the average behaviour of the quantisation error is more adequate to analyse the effect of quantisation error. The error is modelled as a noise, and the second order moment is computed. The noise power metric analyses the dispersion of the finite precision values around the exact value. The noise power metric is adequate when a limited and slight degradation of the application quality due to fixed-point arithmetic can be tolerated. In this case, the system design is based on a trade-off between application quality and implementation cost.

The aim of analytical approaches is to determine a mathematical expression of the fixed-point error metric. The error metric function depends on the word-length of the different data inside the application. The main advantage of these approaches is the short time required for the evaluation of the accuracy metric for a given set of word-lengths. The time required to generate this analytical function can be more or less important but this process is done only once, before the optimisation process. Then, each evaluation of the accuracy metric for a given WL sets corresponds to the computation of a mathematical expression. The main drawback of these analytical approaches is that they do not support all kinds of systems.

Round-Off Noise Power Existing approaches to compute the analytical expression of the quantisation noise power are based on perturbation theory, which models finite precision values as the addition of the infinite precision values and a small perturbation. A quantisation error signal b_i is generated when some bits are eliminated during a fixed-point format conversion (quantisation). This error is assimilated to an additive noise which propagates inside the system. This noise source contributes to the output quantisation noise b_y through the gain α_i. The aim is to define the output noise b_y power expression according to the noise source b_i parameters and the gains α_i between the output and a noise source.

The next paragraphs focus on the model used for the quantisation process, which comprises three phases: (i) *noise generation*, (ii) *noise propagation*, and (iii) *noise aggregation*.

Noise generation. In finite precision arithmetic, signal quantisation leads to an unavoidable error. A commonly used model for the signal quantisation has been proposed in [122]. The quantisation of signal x is modelled by the sum of this signal and a random variable b (quantisation noise). This additive noise b is a uniformly distributed white noise that is uncorrelated with signal x and any other quantisation noise present in the system (due to the quantisation of other signals). This model is valid when the dynamic range of signal x is sufficiently greater than the quantum step size and the signal bandwidth is large enough. This model has been extended to include the computation noise in a system resulting from some bit elimination during a fixed-point format conversion [98, 123, 124].

Noise propagation. Each noise source b_i propagates to the system output and contributes to the noise b_y at the output. The propagation noise model is based on the assumption that the quantisation noise is sufficiently small compared to the signal. Thus, the finite precision values can be modeled by using the addition of the infinite precision values and a small perturbation. A first-order Taylor approximation [125, 126] is used to linearise the operation behavior around the infinite precision values. This approach allows obtaining a time-varying linear expression of the output noise according to the input noise [127]. In [83] and [119], affine arithmetic is used to model the propagation of the quantisation noise inside the system. These models, based on the perturbation theory, are only valid for smooth operations. An operation is considered to be smooth if the output is a continuous and differentiable function of its inputs.

Noise aggregation. Finally, the output noise b_y is the sum of all the noise source contributions. The second order moment of b_y can be expressed as a weighted sum of the statistical parameters of the noise source:

$$E(b_y^2) = \sum_{i=1}^{N_e} K_i \sigma_{b_i}^2 + \sum_{i=1}^{N_e} \sum_{j=1}^{N_e} L_{ij} \mu_{b_i} \mu_{bj} \tag{19}$$

where μ_{b_i} and $\sigma_{b_i}^2$ are respectively the mean and the variance of noise source b_i, and N_e is the total number of error sources. These terms depend on the fixed-point formats and are determined during the evaluation of the accuracy analytical expression. The terms K_i and L_{ij} are constant and depend on the computation graph between b_i and the output. Thus, these terms are computed only once for the evaluation of the accuracy analytical expression. These constant terms can be considered as the gain between the noise source and the output.

These gains can be determined from the impulse response [85] for Linear Time-Invariant systems, or the pseudo impulse response [24] for other smooth systems. In [83, 119], AA is used to keep track of the propagation of every single noise contribution along the datapath, and from this information the coefficients K_i and L_i are extracted. Different hybrid techniques [118, 125, 97] that combine simulations and analytical expressions have been proposed to compute the coefficients K_i and L_{ij} from a set of simulations.

These approaches allow the evaluation of the RON power and are very fast compared to simulation-based approaches. The limit of the proposed methods have been identified. Analytical approaches based on perturbation theory are valid for systems made-up of only smooth operations. For un-smooth operations, hybrid techniques [128] combining simulations and analytical results can used.

7 Conclusion

The efficient implementation of embedded Digital Signal Processing systems requires to test different approximations to explore the trade-off between the implementation cost and the application quality. Data representation in embedded systems is a key aspect to reduce the implementation cost. Fixed-point arithmetic provides low cost and low power mathematical processing capabilities. This low implementation cost is at the expense of a long, tedious and error prone fixed-point conversion process.

In this chapter the two main steps of the fixed-point conversion process are described. The first step corresponding to the integer word-length selection aims at determining the minimal number of bits for the integer part ensuring no overflow. This step is based on the analysis of the dynamic range of each data to evaluate the magnitude upper bound. The second step corresponding to fractional word-length selection aims at determining the minimal number of bits for the fractional part ensuring a sufficient computation accuracy. This step is formulated as an optimization process where the implementation cost is minimised such as the computation accuracy is higher than a minimal constraint. The variables of this optimisation process are the different data word-length. For these two steps, the different existing techniques have been described.

References

[1] Novo D, Farahpour N, Ahmad U, et al. Energy efficient MIMO processing: A case study of opportunistic run-time approximations. In: IEEE/ACM Conference on Design, Automation and Test in Europe; 2014. p. 1–6.

[2] Lapsley P, Bier J, Shoham A, et al. DSP Processor Fundamentals: Architectures and Features. Berkeley Design Technology, Inc; 1996.

[3] Bonamy R, Bilavarn S, Chillet D, et al. Power Consumption Models for the Use of Dynamic and Partial Reconfiguration. Microprocessors and Microsystems. 2014 Jan;.

[4] Novo D, Kritikakou A, Raghavan P, et al. Ultra Low Energy Domain Specific Instruction-set Processor for On-line Surveillance. In: IEEE 8th Symposium on Application Specific Processors (SASP); 2010. p. 30–35.

[5] Woh M, Seo S, Mahlke S, et al. AnySP: anytime anywhere anyway signal processing. In: 36th Annual International Symposium on Computer Architecture. ACM; 2009. p. 128–139.

[6] Dally WJ, Balfour J, Black-Shaffer D, et al. Efficient Embedded Computing. Computer. 2008;41(7):27–32.

[7] Hameed R, Qadeer W, Wachs M, et al. Understanding Sources of Inefficiency in General-Purpose Chips. In: 37th Annual International Symposium on Computer Architecture; 2010. .

[8] Roeven H, Coninx J, Ade M. CoolFlux DSP: The embedded ultra low power C-programmable DSP core. In: Proc. Int. Signal Proc. Conf. (GSPx); 2004. .

[9] Raghunathan V, Raghunathan A, Srivastava MB, et al. High-level synthesis with SIMD units. In: 7th Asia and South Pacific Design Automation Conference (ASP-DAC); 2002. p. 407–413.

[10] TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide; 2006.

[11] Bougard B, De Sutter B, Verkest D, et al. A coarse-grained array accelerator for software-defined radio baseband processing. IEEE micro. 2008;28(4):41–50.

[12] Van Berkel K, Heinle F, Meuwissen PPE, et al. Vector processing as an enabler for software-defined radio in handheld devices. EURASIP Journal on Applied Signal Processing. 2005;16:2613.

[13] Menard D, Chillet D, Sentieys O. Floating-to-fixed-point conversion for digital signal processors. EURASIP Journal on Applied Signal Processing. 2006;14.

[14] Novo D, Bougard B, Lambrechts A, et al. Scenario-based fixed-point data format refinement to enable energy-scalable software defined radios. In: IEEE/ACM Conference on Design, Automation and Test in Europe. ACM; 2008. p. 722–727.

[15] Implementing SIMD in Software; 2006.

[16] Lambrechts A, Raghavan P, Novo D, et al. Enabling WordWidth Aware Energy and Performance Optimizations for Embedded Processors. In: Workshop on ODES; 2007. .

[17] Constantinides GA, Woeginger GJ. The complexity of multiple wordlength assignment. Applied Mathematics Letters. 2002;15(2):137–140.

[18] Constantinides G, Cheung P, Luk W. Wordlength Optimization for Linear Digital Signal Processing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2003;22(10):1432– 1442.

[19] Cantin MA, Savaria Y, Lavoie P. A Comparison of Automatic Word Length Optimization Procedures. In: IEEE Int. Symposium on Circuits and Systems. vol. 2; 2002. p. 612–615.

[20] Catthoor F, de Man H, Vandewalle J. Simulated-annealing-based optimization of coefficient and data word-lengths in digital filters. International Journal of Circuit Theory and Applications. 1988;I:371–390.

[21] Sung W, Kum KI. Word-length determination and scaling software for a signal flow block diagram. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP); 1994. p. 457–460.

[22] Clarke JA, Constantinides GA, Cheung PYK. Word-length selection for power minimization via nonlinear optimization. ACM Trans Des Autom Electron Syst. 2009;14(3):1–28.

[23] Menard D, Serizel R, Rocher R, et al. Accuracy Constraint Determination in Fixed-Point System Design. EURASIP Journal on Embedded Systems. 2008;2008:12.

[24] Rocher R, Menard D, Scalart P, et al. Analytical Approach for Numerical Accuracy Estimation of Fixed-Point Systems Based on Smooth Operations. IEEE Transactions on Circuits and Systems I: Regular Papers. 2012;PP(99):1–14.

[25] Nehmeh R, Menard D, Nogues E, et al. Fast Integer Word-length Optimization for Fixed-point Systems. Journal of Signal Processing Systems. 2015 Mar;p. 1–16. Available from: https://hal.archives-ouvertes.fr/hal-01237217.

[26] Moore RE. Interval analysis. Prentice-Hall series in automatic computation. Englewood Cliffs, N.J.,: Prentice-Hall; 1966.

[27] Neumaier A. Taylor Forms, Use and Limits. Reliable Computing. 2002;9:43–79.

[28] Willems M, Keding H, Grtker T, et al. FRIDGE: An Interactive Fixed-Point Code Generation Environment for Hw/Sw-Codesign. In: Int. Conf. Acoustics, Speed, Signal Processing (ICASSP); 1997. .

[29] Keding H, Willems M, Coors M, et al. FRIDGE: A Fixed-Point Design and Simulation Environment. In: IEEE/ACM Design, Automation and Test in Europe (DATE). Paris, France; 1998. p. 429–435.

[30] Knüppel O. PROFIL/BIAS—A fast interval library. Computing. 1994 Sep;53(3):277–287. Available from: https://doi.org/10.1007/BF02307379.

[31] Revol N, Rouillier F. Motivations for an Arbitrary Precision Interval Arithmetic and the MPFI Library. Reliable Computing. 2005 Aug;11(4):275–290. Available from: https://doi.org/10.1007/s11155-005-6891-y.

[32] Brnnimann H, Melquiond G, Pion S. The design of the Boost interval arithmetic library. Theoretical Computer Science. 2006;351(1):111 – 118. Real Numbers and Computers.

[33] Paruthi V, Mansouri N, Vemuri R. Automatic Data Path Abstraction for Verification of Large Scale Designs. In: IEEE International Conference on Computer Design (ICCD); 1998. .

[34] Lopez JA. Evaluacion de los Efectos de Cuantificacion en las Estructuras de Filtros Digitales Utilizando Tecnicas de Cuantificacion Basadas en Extensiones de Intervalos; Madrid, 2004.

[35] Lee DU, Gaffar AA, Cheung RCC, et al. Accuracy-Guaranteed Bit-Width Optimization. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2006;25(10):1990–2000.

[36] Fang CF, Rutenbar RA, Chen T. Fast, accurate static analysis for fixed-point finite-precision effects in DSP designs. In: Int. Conf. on Computer-Aided Design, 2003 (ICCAD); 2003. p. 275–282.

[37] Cmar R, Rijnders L, Schaumont P, et al. A Methodology and Design Environment for DSP ASIC Fixed Point Refinement. In: IEEE/ACM conference on Design, Automation and Test in Europe (DATE); 1999. p. 271–276.

[38] Carreras C, Lopez JA, Nieto-Taladriz O. Bit-width selection for data-path implementations. In: 12th International Symposium on System Synthesis (ISSS); 1999. p. 114–119.

[39] Benedetti A, Perona P. Bit-Width Optimization for Configurable DSPs by Multi-interval Analysis. In: IEEE Asilomar Conf. on Signals, Systems and Computers; 2000. .

[40] Lopez JA, Carreras C, Caffarena G, et al. Fast characterization of the noise bounds derived from coefficient and signal quantization. In: International Symposium on Circuits and Systems (ISCAS). vol. 4; 2003. p. 309–312.

[41] Comba JLD, Stolfi J. Affine Arithmetic and Its Applications to Computer Graphics. In: VI Brazilian Symp. on Comp. Graphics and Image Processing (SIBGRAPI); 1993. p. 9–18.

[42] Figuereido LHd, Stolfi J. Affine Arithmetic: Concepts and Applications. 10th GAMM/IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics (SCAN). 2002;.

[43] Stolfi J, Figueiredo LHd. Self-validated numerical methods and applications. In: 21st Brazilian Mathematics Colloquium, IMPA; 1997. .

[44] Fang CF, Rutenbar RA, Puschel M, et al. Toward efficient static analysis of finite-precision effects in DSP applications via affine arithmetic modeling. IEEE/ACM Design Automation Conference (DAC). 2003;p. 496–501.

[45] Lee DU, Gaffar AA, Cheung RCC, et al. Accuracy-Guaranteed Bit-Width Optimization. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems. 2006;25(10):1990–2000.

[46] Fang CF, Chen T, Rutenbar RA. Floating-Point Error Analysis Based on Affine Arithmetic. In: IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP). vol. 2; 2003. p. 561–564.

[47] Pu Y, Ha Y. An automated, efficient and static bit-width optimization methodology towards maximum bit-width-to-error tradeoff with affine arithmetic model. In: Asia and South Pacific Design Automation Conf. (ASP-DAC); 2006. p. 886–891.

[48] Lopez JA, Carreras C, Nieto-Taladriz O. Improved Interval-Based Characterization of Fixed-Point LTI Systems With Feedback Loops. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2007;26(11):1923–1933.

[49] Shou H, Lin H, Martin R, et al. Modified Affine Arithmetic Is More Accurate than Centered Interval Arithmetic or Affine Arithmetic. In: 10th IMA International Conference on Mathematics of Surfaces. vol. 2768/2003; 2003. p. 355–365.

[50] Shou H, Lin H, Martin RR, et al. Modified affine arithmetic in tensor form for trivariate polynomial evaluation and algebraic surface plotting. Journal of Computational and Applied Mathematics. 2006;195(1):155 – 171.

[51] Lopez JA, Sedano E, Carreras C, et al. Interval-based analysis and word-length optimization of non-linear systems with control-flow structures. In: The 7th International Conference on Computational Methods (ICCM). vol. 3; 2016. p. 1333–1342.

[52] Esteban L, Lpez JA, Sedano E, et al. Quantization Analysis of the Infrared Interferometer of the TJ-II Stellarator for its Optimized FPGA-Based Implementation. IEEE Transactions on Nuclear Science. 2013 Oct;60(5):3592–3596.

[53] Wu B, Zhu J, Najm FN. Dynamic Range Estimation. IEEE Trans on Computer-Aided Design of Integrated Circuits and Systems. 2006;25(9):1618–1636.

[54] Lopez JA, Caffarena G, Carreras C, et al. Fast and accurate computation of the roundoff noise of linear time-invariant systems. IETCircuits, Devices and Systems. 2008;2(4):393–408.

[55] Wan X, Karniadakis G. An adaptive multi-element generalized polynomial chaos method for stochastic differential equations. Journal of Computational Physics. 2005;209(2):617 – 642.

[56] McClellan JH, Burrus CS, Oppenheim AV, et al. Computer-Based Exercises for Signal Processing Using Matlab 5. Matlab Curriculum Series. New Jersey: Prentice Hall; 1998.

[57] Jeruchim M. Techniques for Estimating the Bit Error Rate in the Simulation of Digital Communication Systems. IEEE Journal on Selected Areas in Communications. 1984;2(1):153–170.

[58] Smith PJ. Quick Simulation: A Review of Importance Sampling Techniques in Communications Systems. IEEE Journal Selected Areas Comm. May 1997;15(4):597–613.

[59] Zeeb CN, Burns PJ. A Comparison of Failure Probability Estimates by MonteCarlo Sampling and Latin Hypercube Sampling. Sandia National Laboratories; 1998.

[60] Kim S, Kum K, Sung W. Fixed-Point Optimization Utility for C and C++ Based Digital Signal Processing Programs. In: Workshop on VLSI and Signal Processing. Osaka; 1995. .

[61] Cantin MA, Savaria Y, Prodanos D, et al. An Automatic Word Length Determination Method. In: IEEE Int. Symp. on Circuits and Systems (ISCAS). vol. 5; 2001. p. 53–56.

[62] Russell KD, Atwood CL, Sattison MB, et al. Saphire Technical Reference Manual: IRRAS/SARA Version 4.0. EG&G Idaho Inc.; 1992.

[63] Tiejten GL. A Topical Dictionary of Statistics. New York: Chapman and Hall; 1986.

[64] Bowker AH, Lieberman GK. Engineering Statistics. Englewood Cliffs, New Jersey: Prentice Hall; 1972.

[65] Coware. Coware SPW. Coware; 2010.

[66] Keding H. Pain Killers for the Fixed-Point Design Flow. Synopsys; 2010.

[67] Mathworks. Fixed-Point Blockset User's Guide (ver. 2.0); 2001.

[68] Eker J, Janneck JW, Lee EA, et al. Taming Heterogeneity, the Ptolemy Approach. Proceedings of the IEEE. 2003;91.

[69] McKay MD, Conover WJ, Beckman RJ. A comparison of three methods for selection values of input variables in the analysis of output from a computer code. Technometrics. 1979;22(2):239–245.

[70] McKay MD. Latin Hipercube Sampling as a Tool in Uncertainty Analysis of Computer Models. In: Swain JJ, Goldsman D, Crain RC, et al., editors. Winter Simulation Conference; 1992. p. 557–564.

[71] Shanmugan KS, Balaban P. A Modified Monte-Carlo Simulation Technique for the Evaluation of Error Rate in Digital Communication Systems. IEEE Trans Commun. 1980;28(11):1916–1924.

[72] Bohdanowicz A. On Efficient BER Evaluation of Digital Communication Systems via Importance Sampling. In: 8th IEEE Symposium on Communications and Vehicular Technology (SCVT); 2001. .

[73] Andradottir S, Heyman DP. On the Choice of Alternative Measures in Importance Sampling with Markov Chains. Oper REs. 1995;43(3):509–519.

[74] Sadowsky JS, Becklew JA. On Large Deviations Theory and Asymptotically Efficient Monte-Carlo Estimation. IEEE Trans Inform Theory. 1990;36:579–588.

[75] Sadowsky JS. On the Optimality and Stability of Exponential Twisting in Monte-Carlo Simulation. IEEE Trans Inform Theory. 1993;39:119–128.

[76] Sakai T, Ogiwara H. Importance Sampling for TCM Scheme over Non-gausian Channel. IEICE Trans Fundamentals of Electron Commun and Comput Sci. 1995;E78A(9):1109–1116.

[77] Orsak GC, Aazhang B. Constrained Solutions in Importance Sampling via Robust Statistics. IEEE Trans Inform Theory. 1991;37:307–316.

[78] Stadler JS, Roy S. Adaptative Importance Sampling. IEEE Journal Selected Areas Comm. 1993;11(3):309–316.

[79] Gumbel EJ. Statistics of Extremes. New York: Columbia Univ. Press; 1958.

[80] Weinstein SB. Theory and Application of some Classical and Generalized Asymptotic Distributions of Extreme Values. IEEE Trans Inform Theory. 1973;19.

[81] Harkness CL, Lopresti DP. Interval methods for modeling uncertainty in ¡e1¿RC¡/e1¿ timing analysis. IEEE Transactions onComputer-Aided Design of Integrated Circuits and Systems. 1992;11(11):1388–1401.

[82] Singhee A, Fang CR, Ma JD, et al. Probabilistic interval-valued computation: toward a practical surrogate for statistics inside CAD tools. In: 43rd ACM/IEEE Design Automation Conference (DAC); 2006. p. 167–172.

[83] Lopez JA, Caffarena G, Carreras C, et al. Fast and accurate computation of the round-off noise of linear time-invariant systems. IET Circuits, Devices and Systems. 2008 august;2(4):393–408.

[84] Sedano E. Automated wordlength optimization framework for multi-source statistical interval-based analysis of nonlinear systems with control-flow structures; Madrid, 2016.

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

[86] Sarbishei O, Radecka K, Zilic Z. Analytical Optimization of Bit-Widths in Fixed-Point LTI Systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2012;31(3):343–355.

[87] Boland D, Constantinides GA. A Scalable Precision Analysis Framework. IEEE Transactions on Multimedia. 2013 Feb;15(2):242–256.

[88] Wadekar SA, Parker AC. Accuracy Sensitive Word-Length Selection for Algorithm Optimization. In: Int. Conf. on Computer Design (ICCD); 1998. p. 54–61.

[89] Caffarena G, Lpez JA, Leyva G, et al. Architectural Synthesis of Fixed-Point DSP Datapaths Using FPGAs. Int J of Reconfigurable Computing. 2009;2009:1–14.

[90] Herve N, Menard D, Sentieys O. Data Wordlength Optimization for FPGA Synthesis. In: Proc. IEEE International Workshop on Signal Processing Systems, (SIPS). Athens; 2005. p. 623–628.

[91] Weijers J, Derudder V, Janssens S, et al. From MIMO-OFDM Algorithms to a Real-Time Wireless Prototype: A Systematic Matlab-to-Hardware Design Flow. EURASIP Journal on Applied Signal Processing. 2006;2006:1–12.

[92] Parashar K, Rocher R, Menard D, et al. A Hierarchical Methodology for Word-Length Optimization of Signal Processing Systems. In: 23rd International Conference on VLSI Design (VLSID); 2010. p. 318 –323.

[93] Han K, Evans BL. Optimum Wordlength Search Using Sensivity Information. EURASIP Journal of Applied Signal Processing. 2006;2006:1–14. Doi:10.1155/ASP/2006/92849.

[94] Caffarena G, Constantinides GA, Cheung PYK, et al. Optimal Combined Word-Length Allocation and Architectural Synthesis of Digital Signal Processing Circuits. IEEE Transactions on Circuits and Systems II. 2006;53(5):339–343.

[95] Fiore PD, Lee L. Closed-form and real-time wordlength adaptation. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). vol. 4; 1999. p. 1897–1900.

[96] Doi N, Horiyama T, Nakanishi M, et al. Minimization of fractional wordlength on fixed-point conversion for high-level synthesis. In: Asia and South Pacific Design Automation Conference (ASP-DAC); 2004. p. 80 – 85.

[97] Fiore PD. Efficient Approximate Wordlength Optimization. IEEE Transactions on Computers. 2008;57(11):1561 –1570.

[98] Constantinides GA, Cheung PYK, Luk W. Truncation Noise in Fixed-Point SFGs. IEE Electronics Letters. 1999;35(23):2012–2014.

[99] Fiore P. A Custom Computing Framework for Orientation and Photogrammetry; 2000.

[100] Caffarena G. Combined Word-Length Allocation and High-Level Synthesis of Digital Signal Processing Circuits; 2008.

[101] Choi H, Burleson WP. Search-based Wordlength Optimization for VLSI/DSP Synthesis. In: IEEE Workshop VLSI Signal Processing; 1994. p. 198 –207.

[102] Sung W, Kum KI. Simulation-Based Word-Length Optimization Method for Fixed-Point Digital Signal Processing Systems. IEEE Transactions on Signal Processing. 1995;43(12):3087–3090.

[103] Han K, Eo I, Kim K, et al. Numerical word-length optimization for CDMA demodulator. In: IEEE International Symposium on Circuits and Systems (ISCAS). vol. 4; 2001. p. 290 –293.

[104] Han K, Evans BL, Swartzlander EE. Data Wordlength Reduction for Low-Power Signal Processing Software. In: IEEE Workshop on Signal Processing Systems; 2004. p. 343–348.

[105] Fang F, Chen T, Rutenbar RA. Floating-Point Bit-Width Optimization for Low-Power Signal Processing Applications. In: IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP). vol. 3; 2002. p. 3208–3211.

[106] Nguyen HN, Menard D, Sentieys O. Novel Algorithms for Word-length Optimization. In: 19th European Signal Processing Conference (EUSIPCO-2011). Barcelona, Spain; 2011. Available from: https://hal.inria.fr/inria-00617718.

[107] Tavsanoglu V, Thiele L. Optimal design of state - space digital filters by simultaneous minimization of sensitivity and roundoff noise. IEEE Transactions on Circuits and Systems. 1984;31(10):884–888.

[108] Thiele L. Design of sensitivity and round-off noise optimal state-space discrete systems. Int J Circuit Theory Appl. 1984;12:39–46.

[109] Berens F, Naser N. Algorithm to System-on-Chip Design Flow that Leverages System Studio and SystemC 2.0.1; 2004.

[110] Kim S, Kum KI, Sung W. 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. 1998 nov;45(11):1455 –1464.

[111] Mentor Graphics. Algorithmic C Data Types; 2008.

[112] Coster LD, Ade M, Lauwereins R, et al. Code Generation for Compiled Bit-True Simulation of DSP Applications. In: IEEE International Symposium on System Synthesis (ISSS); 1998. p. 9–14.

[113] Coors M, Keding H, Luthje O, et al. Fast Bit-True Simulation. In: ACM/IEEE Design Automation Conference (DAC); 2001. p. 708–713.

[114] de Figueiredo LH, Stolfi J. Affine arithmetic: Concepts and applications. Numerical Algorithms. 2004;37(1):147–158.

[115] Alefeld G, Herzberger J. Introduction to Interval Computations. Academic Press, New York; 1983.

[116] Chesneaux JM, Didier LS, Rico F. Fixed CADNA library. In: Conference on Real Number Conference (RNC). Lyon, France; 2003. p. 215–221.

[117] Menard D, Sentieys O. Automatic Evaluation of the Accuracy of Fixed-point Algorithms. In: IEEE/ACM Design, Automation and Test in Europe (DATE). Paris; 2002. .

[118] Shi C, Brodersen RW. A Perturbation Theory on Statistical Quantization Effects in Fixed-Point DSP with Non-Stationary Inputs. In: IEEE Int. Symp. on Circuits and Systems (ISCAS). vol. 3; 2004. p. 373–376.

[119] Caffarena G, Carreras C, Lpez JA, et al. SQNR Estimation of Fixed-Point DSP Algorithms. Int J on Advances in Signal Processing. 2010;2010:1–11.

[120] Barrois B, Parashar K, Sentieys O. Leveraging Power Spectral Density for Scalable System-Level Accuracy Evaluation. In: IEEE/ACM Conference on Design Automation and Test in Europe (DATE). Dresden, Germany; 2016. p. 6.

[121] Constantinides G, Cheung P, Luk W. Roundoff-noise shaping in filter design. In: IEEE International Symposium on Circuits and Systems (ISCAS). vol. 4; 2000. p. 57–60.

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

[123] Menard D, Novo D, Rocher R, et al. Quantization Mode Opportunities in Fixed-Point System Design. In: 18th European Signal Processing Conference (EUSIPCO-2010) (2010). Aalborg, Denmark: EURASIP; 2011. p. 542–546. Available from: https://hal.inria.fr/inria-00534526.

[124] Naud JC, Menard D, Caffarena G, et al. A Discrete Model for Correlation Between Quantization Noises. IEEE Transactions on Circuits and Systems Part 2 Analog and Digital Signal Processing. 2012 Dec;Available from: https://hal.inria.fr/hal-00743413.

[125] Constantinides GA. Word-length optimization for differentiable nonlinear systems. ACM Transactions on Design Automation of Electronic Systems. 2006;11(1):26–43.

[126] Rocher R, Menard D, Scalart P, et al. Analytical accuracy evaluation of Fixed-Point Systems. In: Proc. European Signal Processing Conference (EUSIPCO). Poznan; 2007. .

[127] Menard D, Rocher R, Scalart P, et al. SQNR Determination in Non-Linear and Non-Recursive Fixed-Point Systems. In: European Signal Processing Conference (EUSIPCO); 2004. p. 1349–1352.

[128] Parashar K, Menard D, Sentieys O. Accelerated Performance Evaluation of Fixed-Point Systems With Un-Smooth Operations. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2014 Apr;33(4):599–612. Available from: https://hal.inria.fr/hal-01097606.