System-level approaches for fixed-point refinement of signal processing algorithms

Karthick Parashar

N° d'ordre : ANNÉE 2012

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Traitement du Signal et Télécommunications

École doctorale MATISSE

présentée par

Karthick N. Parashar

préparée à l'unité de recherche INRIA Rennes - Bretagne Atlantique
Institut de recherche en informatique et systèmes aléatoires - CAIRN
École Nationale Supérieure des Sciences Appliquées et de Technologie

---

System-level Approaches for Fixed-point Refinement of Signal Processing Algorithms

Thèse à soutenir à Lannion
Decembre 2012

devant le jury composé de :

CARRERAS Carlos
Professeur à Universidad Politécnica de Madrid / rapporteur

JEGO Christophe
Professeur à IPB/ENSEIRB-MATMECA / rapporteur

CATTHOOR Francky
Professeur à K.U. Leuven / examinateur

JERRAYA Ahmed
Directeur de Recherche, CEA LETI / examinateur

SENTIEYS Olivier
Professeur à l'Université de Rennes 1 / directeur de thèse

MENARD Daniel
Professeur à l'INSA Rennes / co-directeur de thèse

To my loving parents.

Abstract

Fixed-point operators are ubiquitously used for implementation of signal processing systems. A good choice of fixed-point word-length formats leads to savings in the execution cost while not compromising on the accuracy of computation. In spite of good understanding of the quantization effects, it is estimated that about 25%-50% of the design time is still spent on achieving an optimized choice of fixed-point word-length formats in industrial design practices. The word-length optimization problem is known to be combinatorial in nature and arriving at an optimal solution to this problem is known to be NP-hard in complexity, given that every additional optimization variable causes an exponential increase in the combinatorial search space. With many fixed-point operations involved in executing the signal processing algorithm, the scale of the word-length optimization problem can easily grow beyond manageable limits.

In this thesis, a system-level approach is considered for solving the word-length optimization problem, which is a cost minimization problem subject to the system's accuracy performance does not deteriorate beyond a user defined limit. At the heart of this approach is the *divide-and-conquer* technique where, a large system is hierarchically decimated at various levels of sub-system abstractions. The smallest sub-system cost is minimized by solving the classical combinatorial word-length optimization technique. At the system-level, the results from smaller sub-system optimization problems are combined to trade-off performance to minimize the overall cost. Thus, the system-level word-length optimization problem can be thought of as noise-power budgeting problem. The total quantization noise-power generated within a given sub-system is used as an optimization variable instead of the fixed-point operator word-lengths for performing optimizing at the system-level.

The *divide-and-conquer* approach has been made possible as a result of many supporting contributions. The single noise source model (SNS) is a stochastic model which can mimic the effects of quantization noise generated within a signal processing system. This model can be derived analytically with very less simulation effort. Apart from the conventional characterization of quantization noise-power, the noise-power spectral distribution and the noise probability density functions are also analytically estimated as a part of the SNS model. The presence of non-linear operations such as QAM discriminators makes it difficult to analytically estimate the impact of quantization noise at the output of the system. Such operators are classified as *un-smooth operators* contrary to the *smooth* operators whose fixed-point behaviors can easily be captured using simple analytical techniques. The analytical techniques used in the SNS model are applicable to systems with smooth operators only.

In the presence of un-smooth operators, it is inevitable to use fixed-point simulation.

The heavy penalty of time while performing fixed-point simulation is highly undesirable. Keeping this as a motivation, a hybrid simulation technique that accelerates fixed-point simulation by using analytical technique is proposed. This technique uses the *simulate-on-error* strategy. The SNS models are used to capture the fixed-point noise behavior of smooth sub-systems. During simulation, the use of SNS model introduces a statistically equivalent noise at the output of smooth sub-systems thereby reducing the effort for simulation of all smooth sub-systems. The errors due to un-smooth operators are conditional in nature. Occurrence of an un-smooth error can be found on a case by case basis by checking for relevant conditions. Therefore, it is possible to simulate the sub-systems under consideration only in the presence of an un-smooth error. This technique is limited in its application to systems with feed-forward topology. Further, an attempt to analytically arrive at the un-smooth error statistics in the presence of cascading decision operators is made. This effort does not supersede the hybrid technique always as it is only applicable for a certain type of un-smooth operator.

The use of popular heuristics to solve the word-length optimization problem does not guarantee optimality. In this thesis, a convex optimization framework is proposed instead. This framework relaxes the discrete levels of noise-powers that can be assigned to each signal in a combinatorial search space. It considers a convex Pareto-curve to characterize the quantization noise added due to the use of a particular fixed-point operator. The word-length optimization problem can then be cast as a convex cost minimization problem with accuracy constraints. Applying this framework on word-length optimization problems generates optimal points which are not necessarily feasible. However, the optimal point is indicative of the vicinity of global optimal solution. This information is used in the near-optimal word-length optimization algorithm. This information is used to arrive at the optimal or a near optimal solution by quickly conducting a local search.

Contents

List of Figures

Chapter 1

Introduction

The rapid growth of semiconductor technology has fuelled large scale innovation in the electronic product design space. The electronic gadgets available in the market today are not only useful in their own right but there is intense competition between various players for making them efficient, multi-functional and less expensive. In order to keep up with the increasing competition, the product design houses find themselves in a perpetual cycle of design and product delivery. With every iteration of the cycle, the designers need to work with increased technological complexity while they aim to deliver more value for every unit cost they charge from the consumer.

The advances in the telecommunication industry has been able to draw benefits from the growing semiconductor technology. *Smart-phones* of the modern day is a very good example for a complex piece of telecommunication, signal processing and semiconductor engineering. The modern day telecommunication standards such as the fourth generation (4G) communication system technology, wireless communication protocols such as the 802.11x are good examples that demand very high computational power. In all these technologies, the implementations are expected to comply with the performance demand of both voice and data traffic. In case of the *smart-phone*, many such algorithms are implemented on a single device. Moreover, they are also loaded with various multimedia features for recording and playing back music and video in real time. The fact that all this would happen under strict energy, form-factor and time constraints makes it a remarkable piece of engineering.

Drawing from the *smart-phone* example, it may be generalized that the design of any modern day electronic gadget has to be such that the system cost which is usually measured in terms of silicon area, power profile and the execution time is kept to a minimum while not compromising on the system performance expressed in terms of various metrics such as computational accuracy and the response time. Very often, these goals are conflicting in nature and the designer has to inevitably make a trade-off between the system performance and its cost. Therefore, it is very important to make careful choices in every design step to ensure the best possible performance of the entire system. The choice of operators used to implement these algorithms has a large impact on the cost-performance trade-off. Floating-point and

fixed-point operators are two popular choices available for the implementation of all arithmetic operations. Among them, the fixed-point arithmetic operators are known to take significantly lesser area, shorter latency and are known to consume lesser power. Implementation of telecommunication algorithms usually have rigorous performance parameters to be achieved and demand high computational power. In such cases, the use of fixed-point operations is a popular and accepted choice.

This thesis addresses the problem of assigning optimal fixed-point number formats to operations for the implementation of a given signal processing system keeping in mind both correct functionality and being able to meet the rigorous performance parameters. In this chapter, an attempt is made to describe the genesis of this problem. In the following sections, the problem of fixed-point refinement is presented in the context of a system design flow and the actual problem addressed in this thesis is formally stated. The contents of the thesis and its structure are briefly discussed and finally the contributions from this thesis are outlined.

Fixed-Point Refinement

The process of electronic system design consists of many steps and the designer is expected to make the right choice considering multiple options available in each of these steps. The advantages of a particular choice which improves the system performance would invariably be associated with an increase in the system cost which is undesirable. Therefore, making the right choice usually requires the designer to go through a number of iterations before arriving at an optimal choice.

Consider the task of implementing a given signal processing algorithm. The various steps for executing this task is as given in Figure 1.1. The first block is the design of the algorithm itself and in the context of this thesis, it is a given. The very next step is the determination of suitable fixed-point representation. Even if optimality is not a concern, at least an arbitrary assignment of fixed-point formats is a must while considering its implementation on modern computational platforms. This activity is the main focus of the thesis. Once a suitable fixed-point format is decided, the system is implemented and it is checked if the performance of the system meets the user defined criteria and satisfies the constraints. If the results are satisfactory, the system design is signed-off. Otherwise, learning from the inadequacies, all the steps starting from fixed-point refinement has to be repeated to compensate for the same.

The *Implementation* block is just shown as yet another step among the rest of the things in the design flow. This is done in order to keep the focus on the problem considered in this thesis. In practice, this block is complex and could be broken down into many other similar tasks that solve equally important and difficult optimization problems. Implementation of a fixed-point design on software and hardware platforms has been studied extensively.

In case of a software implementation platforms, there are optimizing compilers dedicated for specific processor architectures. The modern day compiler infrastructure provides many options to generate optimal machine code (for e.g. GCC optimiza-

tion flags [46]) tuned for reducing memory and time of execution. The success of the SIMD[1] technology has brought in the concept of auto-vectorization where vectorized code is generated automatically by observing data-parallel loops. A good fixed-point representation can impact the choices made in the later stages.

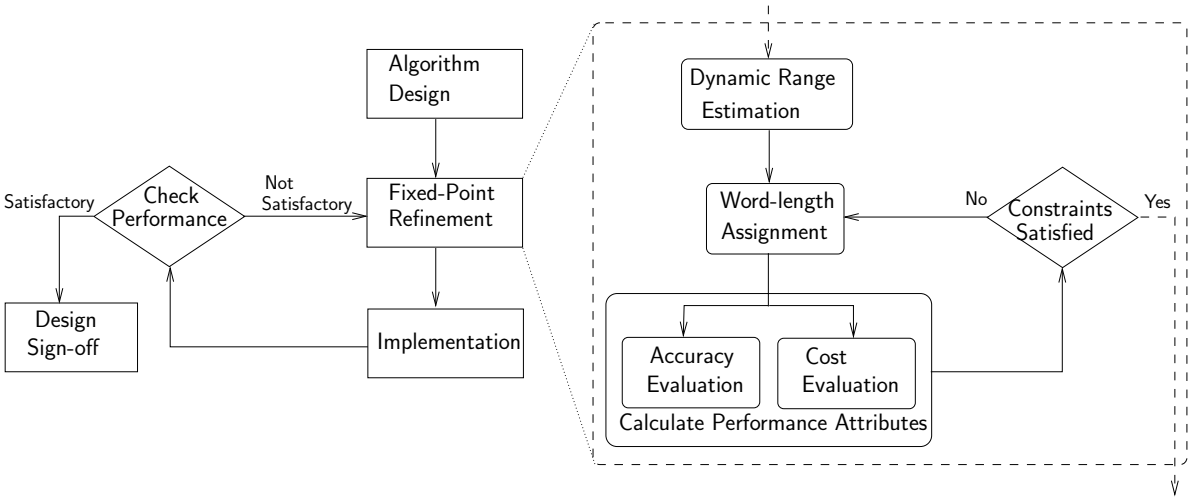


Figure 1.1: Electronic System Design Flow

In case of hardware implementation platforms, a number of high-level synthesis tools such as *Bluespec* [8], *Impulse C* [40] that are commercial and tools such as *C-to-Verilog* that are open-source exist for automatic realisation of sequential algorithmic descriptions into synthesizable RTL[2] designs. Given the ease with which fixed-point arithmetic can be implemented using digital hardware, it is natural to use fixed-point number representations for hardware implementation. A good fixed-point number format is therefore inevitable even before an algorithm is presented to such an automatic high-level synthesis tool.

For any given system that needs to be implemented using fixed-point arithmetic, there are four sub-tasks as shown in Figure 1.1 that essentially solve one aspect of the fixed-point refinement process. The first sub-task is that of estimating the dynamic range of the signals. The second is that of measuring the loss in accuracy due to the usage of fixed-point numbers and operators. The third problem is that of estimating the corresponding cost benefits for using a given fixed-point format. The fourth problem is to balance the loss in accuracy and the gain in system implementation costs while respecting the design constraints by making an optimal or at least an optimized choice of fixed-point format. It has been shown that the word-length optimization problem is an NP-hard [34] combinatorial problem. Usually, such problems are solved by using heuristic guidelines iteratively. Every iteration during the optimization process can be very time consuming as it requires the evaluation or estimation of computational accu-

[1]Single Instruction Multiple Data
[2]Register Transfer Level

racy and the total cost every time. In [26], a large number of industrial implementation efforts were studied and it has been reported that nearly $25\% - 50\%$ of the designer's effort is spent in determining the fixed-point representation for the system. A broad overview of techniques for addressing each of these problems is discussed in detail in Chapter 2.

The Word-length Optimization Problem

The trade-off between the fixed-point word-lengths and the cost incurred can be expressed in the canonical form as

$$\min\left(C\left(\mathbf{w}\right)\right) \quad \text{subject} \quad \text{to} \quad \lambda\left(\mathbf{w}\right) \geq \lambda_{obj}, \tag{1.1}$$

where $\mathbf{w}$ is a vector of fixed-point word-lengths of various fixed-point operations in the system. Each element w_j in this vector is an optimization variable. There are as many w_j as the number of operations in the system. $C(\mathbf{w})$ is the cost function which computes a relevant cost metric as a function of different word-length choices. $\lambda(\mathbf{w})$ is the accuracy evaluation function which is also a function of assigned fixed-point word-lengths and λ_{obj} is the accuracy constraint. The inequality in the optimization constraint is dependent on the metric used for measuring the accuracy. If SQNR[1] is used to quantify the accuracy, it is expected that the metric is higher than a certain minimum and hence the inequality becomes *greater-than-or-equal*. On the other hand if BER or the total quantization noise power is used to quantify accuracy, the user would set an upper bound on the value and hence the inequality becomes *lesser-than-or-equal*.

While this problem has been addressed by many approaches in the past, this thesis focuses on the scalability of such approaches with growing system sizes. As it will be discussed in the rest of the thesis, the lack of scalability of the existing optimization techniques proves to be an impediment in applying many of the techniques on large systems.

In order to address the scalability issues, a *divide-and-conquer* approach that can scale well with growing system sizes is proposed to solve the word-length optimization problem in this thesis. In this approach, the system-level word-length optimization problem is broken down into smaller sub-problems with each problem addressing the word-length optimization of a constituent sub-system. Each of these sub-problems is assigned an accuracy budget measured by quantization noise-power. The sub-problems attempt to meet the budgeted accuracy constraint while minimizing the sub-system costs. The system-level word-length optimization problem is then a function of the accuracy of the sub-system. The optimization problem can then be written as

$$\min\left(C\left(\mathbf{q}\right)\right) \quad \text{subject} \quad \text{to} \quad \lambda\left(\mathbf{q}\right) \leq \lambda_{obj}, \tag{1.2}$$

where $\mathbf{q}$ is the vector of accuracy of each of the sub-systems. If the system consists of

[1] Signal to Quantization Noise Ratio: the ratio of signal power and quantization noise power expressed in decibels (dB)

N sub-systems, $\mathbf{q} = [q_1, q_2, \ldots, q_N]$. The system cost C and the total system accuracy measure λ is a function of $\mathbf{q}$.

The sub-system optimization problems can be decomposed further into smaller sub-system optimization problems. This continues until the problem size is small enough to be optimized as a classical word-length optimization problem. The sub-problem corresponding to the sub-system word-length optimization can then be written as

$$\min\left(C_i\left(\mathbf{w}^i\right)\right) \quad \text{subject} \quad \text{to} \quad \lambda_i\left(\mathbf{w}^i\right) \leq q_i, \tag{1.3}$$

where $\mathbf{w}^i$ is the word-length vector of fixed-point operations in the i^{th} sub-system. $C_i(\mathbf{w}^i)$ and $\lambda_i(\mathbf{w}^i)$ are the corresponding cost and accuracy evaluation functions. q_i is the accuracy evaluation metric expected out of the i^{th} sub-system. It has to be noted here that the vector $\mathbf{w}^i$ is a sub-set of the vector $\mathbf{w}$.

Thesis Contributions

The central idea in this thesis rallies around the ability to analytically estimate the quantization noise power and some of its characteristics at the output of a given signal processing system. The contributions in this thesis add to the state of the art on various ways of using the quantization noise characteristics for fixed-point performance analysis and word-length optimization problems. These contributions can be broadly classified into three categories.

The Single Noise Source Model

The quantization noise characteristics at the operator is modeled as a random process and its characteristics are well understood. The single noise source (SNS) model extends this understanding to the system-level. The total quantization noise at the output of a system can be modeled as a sum of various random processes. Apart from the knowledge of quantization noise power, it is also important to understand its spectral and distribution functions to accurately characterize the random process.

In this thesis, analytical techniques to derive the spectral power density and probability density functions of the quantization noise at the output of a given system are derived. Calculation of these parameters can be computationally intensive. However, the spectral and probability density functions are required to be evaluated only under certain conditions. Therefore, it continues to remain white with a Gaussian distribution in general unless mentioned otherwise. A practical approach is therefore to selectively calculate these parameters only when they are required. An algorithm to identify such scenarios from a system-level graph is described.

Un-smooth Quantizers

The analytical models describing the quantization noise are based on Widrow's quantization models and perturbation theory. The models are accurate only when the

quantization step size is very small in comparison to the dynamic range. As the quantization step-size increases, the noise properties deviate from analytical predictions and soon become intractable. Such quantizers are referred to as un-smooth quantizers.

In this thesis, an algorithm to identify such un-smooth quantizers is proposed. An analytical technique for the propagation of quantization noise in the presence of un-smooth operator is also proposed. In cases when more than one un-smooth operators are present, it is important to consider the effect of previous quantization errors in order to calculate the error at the output of an un-smooth quantizer. The experiments for calculation of quantization errors in presence of previous un-smooth errors was carried out in collaboration with Aymen Chakari during the preparation of his doctoral thesis.

Apart from un-smooth quantizers, there can be other operators whose behavior in response to perturbation by quantization noise cannot be captured by analytical formula. Also, the technique for propagation of error statistics through the system in presence of un-smooth operators is limited to systems with feed-forward topology. To address this scenario, a hybrid technique to accelerate the fixed-point simulation which takes the benefit of analytical techniques by the application of the SNS model is proposed. The acceleration obtained as a result of this approach can potentially decrease the total simulation time by several orders of magnitude.

Hierarchical Word-length Optimization

Fixed-point word-length optimization can be broadly approached in two directions depending upon the optimization criteria. It could either be a cost minimization problem under an accuracy constraint or an accuracy maximization problem under a given cost constraint. In general, the base-line principle for any product design is that the end product meets the design expectation. Improvement in its performance is generally secondary to its functional correctness. Keeping this point of view, the cost minimization problem rather than the accuracy maximization problem is the subject of this thesis.

Word-length optimization problem is NP-hard in its complexity. As the system sizes grow, the combinatorial search space grows exponentially with every new optimization variable. All the heuristics proposed to solve this problem work with fixed-point word-lengths. In this thesis, the word-length optimization problem is viewed as a system-level quantization noise budgeting problem. To begin with, an adaptation of an existing greedy word-length optimization algorithm for budgeting the total quantization noise. Some approximations on the trade-off between quantization noise and the cost of implementation are made such that the trade-off becomes convex. This relaxation enables exploitation of the linear quantization noise propagation properties and the system-level noise-budgeting problem is cast as a convex optimization problem. The solution obtained by solving the convex optimization problem is used to perform a local search for suitable word-length assignments which can lead to solutions very close to optimality. The *near-optimal* word-length optimization algorithm essentially proposes a polynomial time heuristic algorithm to solve the otherwise NP-hard problem.

Thesis Organization

The contents of this thesis is organized in six chapters. This introductory chapter introduces the central theme and organization of this thesis. The second chapter presents a survey of the relevant literature comprising of accuracy evaluation and word-length optimization. It also outlines the challenges involved in performing fixed-point refinement of large signal processing systems with existing techniques.

The third chapter focuses on the problem of hierarchical accuracy evaluation of fixed-point systems. The concept of the single noise source (SNS) model is developed in this chapter as a solution for reducing the complexity of estimating the loss in accuracy of a system hierarchically. The various subsections are devoted to the hierarchical accuracy evaluation methodology and analytical determination of the SNS model parameters.

The fourth chapter focuses on the un-smooth quantizers and other operators whose response to perturbation by quantization noise is not captured analytically. This chapter takes a fresh look at the definition of smoothness of quantization noise. Other sub-sections provide an analytical technique for estimation of probability of error and hence the total quantization noise power at the output of an un-smooth quantizer. This chapter also describes the *hybrid* simulation algorithm for acceleration of fixed-point simulation in the presence of un-smooth operators.

The fifth chapter focuses on the word-length optimization. It first presents a *divide-and-conquer* strategy for performing hierarchical word-length optimization. This algorithm is a simple adaptation of the greedy *Min +1 bit* algorithm. From the view point of noise-budgeting, this problem is cast as a convex optimization problem and an alternate *near-optimal* word-length optimization algorithm is proposed.

The sixth chapter summarizes the impact of this work and prospects future directions to take this work forward.

List of Publications

Some of the results obtained during the preparation of this thesis were presented at various peer reviewed international conferences. A list of publication is presented in the reverse chronological order (starting from the latest).

- A. Chakari, K. Parashar, R. Rocher, P. Scalart, "Analytical approach to evaluate the effect of the spread of quantization noise through the cascade of decision operators for spherical decoding", to be presented at *Conf. on Design and Architectures for Signal and Image Processing, DASIP 2012*, Karlsruhe, Germany, Oct. 2012

- K. Parashar, D. Menard and O. Sentieys, "Approche hiérarchique pour l'optimisation de la précision des systèmes de traitement du signal utilisant l'arithmétique virgule fixe," in *Proc. of GRETSI 2011*, Bordeaux, France, Sep. 2011.

- K. Parashar, D. Menard, R. Rocher and O. Sentieys, "Shaping Probability Density Function of Quantization Noise in Fixed point Systems," in *Proc. of Asilomar Conference on Signals, Systems and Computers, AsilomarSSC 2010*, Asilomar, California, USA, pp 1675-9, Nov. 2010.

- K. Parashar, D. Menard, D. Novo, R. Rocher, O. Sentieys and F. Catthoor, "Fast performance evaluation of fixed-point systems with un-smooth operators," in *Proc. of IEEE/ACM International Conference on Computer Aided Design, ICCAD 2010*, San Jose, California, USA, pp. 9-16, Nov. 2010.

- K. Parashar, D. Menard, R. Rocher and O. Sentieys, "Estimating Frequency Characteristics of Quantization Noise for Performance Evaluation of Fixed Point Systems," to appear in *Proc. of European Signal Processing Conference, EUSIPCO 2010*, Albourg, Denmark, pp. 552-6, Aug. 2010.

- K. Parashar, R. Rocher, D. Menard and O. Sentieys, "Analytical Approach for Analyzing Quantization Noise Effects on Decision Operators," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2010*, Dallas, USA, pp. 1554-7, Mar. 2010.

- K. Parashar, D. Menard, R. Rocher and O. Sentieys, "A hierarchical Methodology for Word-Length Optimization of Signal Processing Systems," in *23rd International Conference on VLSI Design, VLSID 2010*, Bangalore, India, pp. 318-23, Jan. 2010.

Chapter 2

Background and Previous Work

Fixed-point and floating-point number formats are two popular binary arithmetic number formats available on modern day computing platforms. Fixed-point arithmetic circuits have existed from the earliest of the days ever since electronic computers and calculators have been in use for computations. Choosing the right format of fixed-point arithmetic has always been a matter of concern while working with computers to solve numerical problems. In the early days, the choice of fixed-point arithmetic format was influenced mostly by memory considerations. Back then, memory was both slow and expensive and the computational power of the processors were also very low. Most complex numerical algorithms had to depend on precomputed look-up tables during computations. Today, the definition of fixed-point number formats is impacted by considerations such as the silicon area, the execution time and the total energy consumption of the electronic device.

The problem with fixed-point arithmetic has always been that the number of bits have to be traded-off between the dynamic range and precision of the number. By using the floating-point format for number representation, both problems of high dynamic range and high precision are solved. However, this solution is comes at the cost of increased complexity of arithmetic circuit in comparison with fixed-point arithmetic circuits. Owing to its complexity, the floating-point arithmetic operations were traditionally available as a co-processor system package. This was also the case because, the need for using floating-point units seldom arise for performing the tasks the early computers were classically required to. Indeed, it was for this reason that the fixed-point arithmetic units were preferred over floating-point units for integration into the processor data path. As the processors became more powerful and more energy efficient, applications such as multimedia and communication were deployed while the basic premises of the processor architecture was kept intact for purposes such as backward compatibility. In all such attempts, the implementation of complex signal processing algorithms continued using fixed-point arithmetic data-paths for computations. Such algorithms are complex and demand high computational power and numerical accuracy. Although floating-point arithmetic units provide very high accuracy, their use for purposes of implementing complex signal processing algorithms on embedded systems

is inefficient due to the large area, long latency and high power consumption costs even today.

To implement a given signal processing algorithm on fixed-point processing platforms, the numerical aspects of given algorithms has to be suitably tuned such that the limited precision or limited dynamic range of the fixed-point format does not affect the performance of the algorithm. The development of computer architecture broadly reflects the increased trend of using fixed-point operations. Induction of the multimedia extension instruction set (MMX) in Intel's architectures and the availability of a number of processors with wide *SIMD*[1] capable data-paths belonging to the family of popular digital signal processors are available off the shelf. These have been used extensively for implementation of various signal processing applications. Likewise, the fixed-point arithmetic circuits are sought out instead of floating-point circuits for implementation of signal processing algorithms owing to the power dissipation and area concerns. Therefore, the problem of fixed-point refinement which is also popularly referred to as the float-to-fix conversion problem focusses on using available fixed-point data paths for implementation of signal processing algorithms.

The fixed-point refinement problem is often addressed with the aim of reducing power consumption, reduction in circuit area and efficient re-timing on ASIC[2] and FPGA[3] implementations. Also, power consumption and execution time factors are increasingly being considered for optimization on micro-processor platforms. With the availability of reconfigurable processor data-paths such as the SIMD, the choice of fixed-point data types cast their influence on the total power consumption and execution time of software. Consequently, the focus has shifted from realising just an acceptable fixed-point implementation to an optimized fixed-point implementation of the system.

The various activities under the fixed-point refinement process has been classified into four sub-tasks as described in Figure 1.1 in the previous chapter. Each of these sub-tasks are non-trivial and seek extensive research efforts. Although the problem of fixed-point refinement seeks a solution to each of the sub-tasks, they cannot be considered simultaneously due to the enormity of the issues involved. For example, it is hard to capture the errors caused due to inadequate assignment of dynamic range and in such circumstances, it is difficult to separate the effect of insufficient precision causing overflow or saturation errors. The characteristics of such errors cannot be easily captured with closed form expressions. In a practical step-by-step approach, the dynamic range is determined to begin with. With sufficient bits assigned to the integer part, the precision part of the fixed-point format is optimized to reduce costs. In this chapter, various existing techniques for accomplishing each of these tasks are considered. This collage of information is used to motivate the need for improved techniques presented in this thesis.

[1]SIMD: Single Instruction Multiple Data
[2]ASIC: Application Specific Integrated Circuit
[3]FPGA: Field Programmable Gate Array

2.1 Computation with Binary Arithmetic

Binary number representation is native to almost all digital platforms today. The use of radix-2 or binary arithmetic is therefore a natural choice. Floating-point and fixed-point representations are two popularly used number formats for the purpose of storage and computation. In general, signal processing algorithms are known to be computationally intensive. For the purpose of efficient implementation of any signal processing algorithm, the choice of number format plays a very crucial role. Therefore, it is important to consider both formats before a decision is made.

2.1.1 Fixed-Point Number Format

The layout of the fixed-point number format is as shown in Figure 2.2.

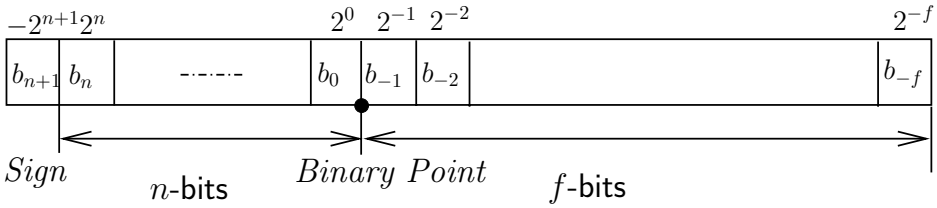


Figure 2.1: Fixed-point number format

This is a signed number format using the 2's complement place value system. A 1 in the most significant bit represents a negative number. The value of a string of bits in this format v_{fi} is given as

$$v_{fi} = v_{bin} \times 2^{-f}, \tag{2.1}$$

where v_{bin} is the integer value of the 2's complement number from bits 0 through $n + f + 1$ (MSB[1] through LSB[2]). This number format consists of one sign bit, n bits for the integer part and f bits for representation of the fractional part. This is also popularly referred to as $n.f$ format indicating the integer and fractional bits used for the fixed-point representation. In a fixed-point system with the $n.f$ format applied universally, every signal value is represented using $(n+f+1)$ bits. The decimal number 0.1 in binary is written as

$$0.1_{(10)} \quad = \quad 0.000\overline{1100}_{(2)}. \tag{2.2}$$

The $\overline{1100}$ indicates that these bits keep recurring infinitely on the right hand side of the binary number. Consider representing this number in the $n.f$ format which consists of one sign bit, 3 integer bits and 6 fraction bits. Due to the limited precision in the fraction bits, the actual number represented is close to 0.1 but is not exactly equal to

[1]MSB: Most Significant bit
[2]LSB: Least Significant Bit

the value of 0.1. Considering the fixed-point error, the number 0.1 is written in the 3.6 format as

$$\begin{aligned} 0.1_{10} &= 0000.000110_2 + 0.00625_{10} \\ &= \underbrace{0}_{s-bit}\ \underbrace{000}_{n-bits} \ . \ \underbrace{000110}_{f-bits} + 0.00625_{10}. \end{aligned} \tag{2.3}$$

The binary position is virtual and decided by the user who designs the number format for the given application. In the above example, since the integer part is set to 0, all the 9 bits could have been assigned to the fractional part thereby increasing the precision of the number. However, this choice makes it impossible to accommodate any number whose magnitude is greater than 1. In other words, the number of bits assigned to the dynamic range is at the cost of loss of precision and vice-versa unless more bits are added to the number representation on the whole. A good fixed-point format achieves the right balance between the number of integer and fractional bits while keeping the total number of bits to the minimum.

2.1.2 Floating-Point Number Format

The floating-point number format is also a signed number representation format based on the 2's complement representation. The layout of the floating-point number format is as shown in Figure 2.2. One bit is dedicated to represent the sign of the number.

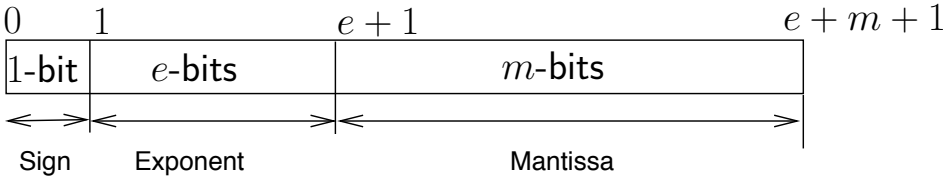


Figure 2.2: Floating point number format

The numbers represented in this format are normalized to a leading 1 in the binary representation and *mantissa* number of significant bits to the right hand side of the leading 1 are stored in the mantissa part. The number of shifts required to do the normalization of the binary representation of the number to the leading 1 is coded using the 2's complement notation in the *exponent* bits. A series of bits represented in this format is interpreted as the value v_{fl} that can be computed as

$$v_{fl} = \{-1\}^s \times \{1 + m_{bin} \times 2^{-m}\} \times 2^{e_{bin}-b}, \tag{2.4}$$

where m_{bin} and e_{bin} are the value of *mantissa* and *exponent* bits respectively coded in binary with m bits and e bits respectively. The mantissa is positive as the sign of the number is already taken care of by the sign bit. Depending upon the magnitude of the number being encoded, it is required to shift the binary representation either

to the left (if magnitude < 1) or to the right (if magnitude > 1). Here, shifting left amounts to multiplying the normalized number with negative powers of 2 and shifting right corresponds to multiplying the normalized number with positive powers of 2. However, the *exponent* filed takes on un-signed positive integer values. Therefore the exponential bias b is subtracted from the actual exponent value to include multiplication with negative powers of 2.

A number of different possible floating-point number formats depending upon the choices of number of bits assigned for the mantissa, exponent and the bias. However, two popular floating point number formats are defined as a part of the IEEE floating point specification. The single precision floating point format has 8 bits assigned for the exponential part and 23 bits for the Mantissa. The double precision format assigns 11 bits for the exponent and 52 bits for the mantissa part. Thereby, the single and double precision formats are 32 bits and 64 bits wide respectively.

Consider representing the decimal number 0.1 in the binary format. The binary value of this number is given as

$$\begin{aligned} 0.1_{(10)} &= 0.000\overline{1100}_{(2)} \\ &= \{-1\}^0 \times 2^{-4} \times 1.100\overline{1100}_{(2)}. \end{aligned} \tag{2.5}$$

Normalization of this number to the leading 1 in its mantissa part requires a left shift by 4 binary places. Post normalization, the number of binary digits encoded in this number can be only as long as m bits. Storing a finite number of bits leads to loss of precision of the number and it is inevitable. It has to be noted here that the $1.m$ format removes maximum number of leading zeros and hence provides at least one extra bit of precision than if the $0.m$ format were to be used. In this example, the exponent value needs to represent multiplication with 2^{-4}. The bias is set to 127 and 1023 respectively for single and double precision formats. The binary value of the *exponent* is obtained by adding the bias value of the format to the actual power of 2 with which the normalized number needs to be multiplied with.

The floating point number is designed to represent small fractions as well as numbers with large magnitude that would occur during scientific computations. In other words, the number representation must have both a large dynamic range and very good precision in its representation. Given the number of *exponent* bits, the bias is set to the median of the range of values that are represented by the bits. Doing so ensures that the smallest fraction is as many orders of magnitude smaller than 1 as much as the order of magnitude of the largest number greater than 1. In this format, representation of a negative number is trivial as it is sufficient to flip the sign bit alone.

2.1.3 Computation with Fixed-Point and Floating-Point numbers

From the above discussion, it is clear that the floating point number format is better than the fixed-point format in terms of providing enough dynamic range and precision. However, this is achieved at the cost of complexity of the arithmetic operators. Addition and multiplication are two most commonly used operations. The costs of performing

these operations is therefore considered in this section.

Addition

The fixed-point number is natively in the binary format. Any two fixed-point numbers can be added together using a simple ripple carry adder circuit after aligning the binary points of both numbers. The total effort for fixed-point addition ς^a_{fix} is equal to the effort for binary addition ς_a. This is written as

$$\varsigma^a_{fix} = \varsigma_a. \tag{2.6}$$

On the other hand, the floating point number is in a packed format. First it has to be un-packed to expose the sign, exponent and mantissa. Then the exponent has to be normalized by comparing the bits in the exponent part of both numbers and shifting the mantissa bits such that the exponents are now equal. If the sign bit is set, the mantissa bits are converted to its 2's complement form before addition. After addition, the result of the addition has to be converted to its 2's complement form if it is found to be negative. Clearly, the addition of two fixed-point numbers is relatively easier than adding two floating point numbers. If ς_{cmp} is the effort required for comparison of the exponents and ς_{com} is the effort required for calculating the 2's complement of the number, the total effort for floating point addition ς^a_{flt} in the worst case is given as

$$\varsigma^a_{flt} = \varsigma_a + \varsigma_{cmp} + 2.\varsigma_{com}. \tag{2.7}$$

It is known that the efforts ς_a, ς_{cmp} and ς_{com} are all of the order $O(N)$ where N is the number of bits assigned to the output of the adder. The time for un-packing the floating point format is ignored as it is trivial in hardware.

Multiplication

A hardware multiplier circuit that can multiply two signed binary numbers could be used to multiply two fixed-point numbers. The result of a multiplication has more number of bits both in its dynamic range and its precision. The binary point position of the resulting fixed-point number is determined by taking into account the binary point positions of the two numbers that were multiplied. Usually, a number of least significant bits of the result are discarded either by various truncation or rounding modes to obtain the final result. Therefore, the total effort for fixed-point multiplication ς^m_{fix} is ς_m: the time taken for binary multiplication. This is written as

$$\varsigma^m_{fix} = \varsigma_m. \tag{2.8}$$

For multiplying two floating point numbers, their exponents are simply added and the mantissa and the sign bit are multiplied. The total effort for floating point multiplication ς^m_{flt} is given as

$$\varsigma^m_{flt} = \varsigma_m + \varsigma_a. \tag{2.9}$$

The multiplier algorithm can be as high as $O(N^2)$ and the time for addition is $O(N)$ where N is the number of bits assigned to the output word-length of the multiplier.

2.1.4 Choice of Number Representation

A good choice for representing numbers must be able to support a wide dynamic range as well as fine precision. In other words, the number representation gets more versatile with higher ratio of dynamic range to the precision. Typically, both floating-point and signed fixed-point number representations are symmetric in their ranges. Therefore, the dynamic range ratio D, defined as the ratio between the largest and smallest numbers that can be used for measuring the fidelity of the number representation system. The dynamic range ratio is also expressed in decibels (dB) by taking 20 times the log_{10} of the dynamic range. The dynamic range ratio and its equivalent in decibels is defined as

$$\begin{aligned} D &= \frac{max(|x|)}{min(|x|)} \\ D_{dB} &= 20 \times log_{10}(D), \end{aligned} \tag{2.10}$$

where $|x|$ represents the magnitude of the number x and the functions $max(|x|)$ and $min(|x|)$ respectively represent the maximum and minimum magnitude of values taken by the number x excluding the value of 0.

Let b bits be assigned to the fixed-point number and n, f be the number of bits assigned to the integer and fractional bits. The dynamic range is therefore given as

$$\begin{aligned} D &= \left|\frac{2^{n-1}}{2^{-f}}\right| \\ &= \left|\frac{2^{n+f-1}}{1}\right| \\ D_{dB} &= 20 \times log_{10}(2) \times (b-1) \\ &= 6 \times (b-1). \end{aligned} \tag{2.11}$$

So, addition or omission of one least significant bit can improve or depreciate the quality of the numbers represented by $6dB$.

In case of the floating-point number representation, the total number of bits is split among the exponent and the mantissa in the ratio of $1:3$. Though this is not a hard and fast rule, this is the usual assumption for floating point numbers. The ratio of the

power of the largest and the smallest positive numbers is written as

$$\begin{aligned} D &= \frac{2^{e_{max}} \times (2 - 2^{-m})}{2^{e_{min}} \times (1 + 2^{-m})} \\ &\simeq \frac{2^{e_{max}} \times 2}{2^{e_{min}}} \\ D_{dB} &= 20 \times log_{10}(2) \times (e_{max} - e_{min} + 1) \\ &= 20 \times log_{10}(2) \times 2^{\frac{b}{4}} \\ &= 6 \times 2^{\frac{b}{4}}. \end{aligned} \tag{2.12}$$

Comparing Equations 2.11 and 2.12, it is clear that the dynamic range ratio expressed in decibels D_{dB} increases exponentially in case of floating-point numbers and linearly in case of fixed-point numbers with increasing number of bits assigned. The dynamic range for various bit widths assigned for number representation in both cases is plotted in Figure 2.3. When the number of bits b is small, the dynamic range of the floating point number is much inferior to fixed-point numbers. With increasing number of bits, the dynamic range ratio of the floating point representation turns out to be better than the fixed-point representation.

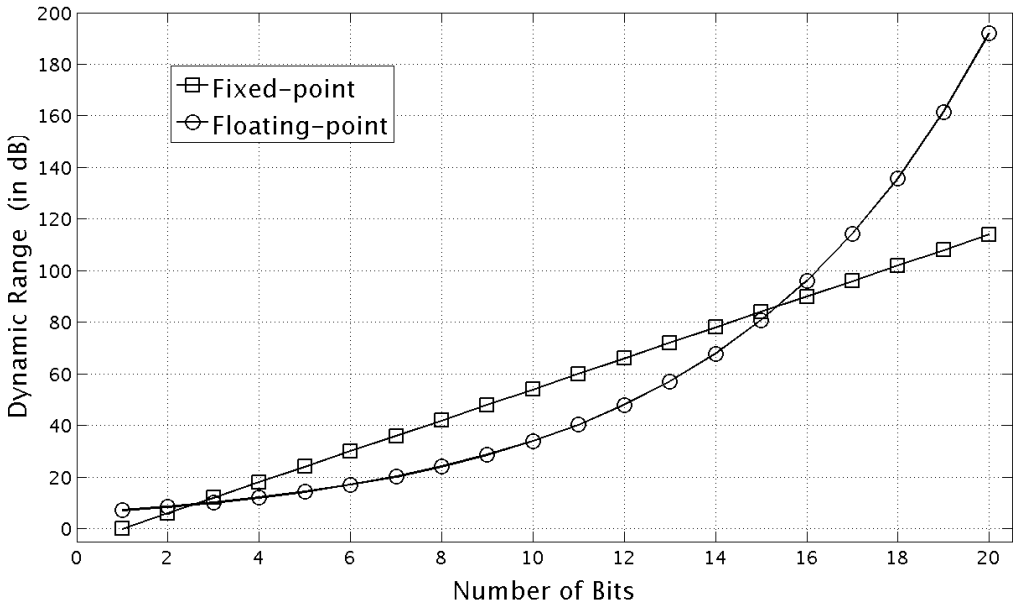


Figure 2.3: Dynamic range ratio of fixed-point and floating-point number formats

Under the consideration of designing floating-point numbers with the ratio of 1 : 3 for the exponents and mantissa bits. Until 16 bits, the dynamic range of the fixed-point representation is better than the dynamic range of the floating-point number representation. The floating-point number representation takes off exponentially far

exceeding the dynamic range of fixed-point numbers.

Another consideration for making the choice between the two formats is the increase in the maximum error with increasing dynamic range of the signal. Consider a given number of bits assigned for representing the given number. In case of fixed-point representation, more bits have to be assigned for the integer part to accommodate the growing dynamic range. This reduces the number of bits in the fractional part and hence contributes to the decrease in the signal to quantization noise ratio (SQNR) of the quantized signal. In case of floating-point representation, with increasing dynamic range the exponent is incremented while the mantissa bits are a constant. Thereby, the quantization step-size increases in proportion with the increasing dynamic range. Therefore, the ratio of the SQNR is almost a constant. If the dynamic range of the signal is small enough, the fixed-point numbers can be designed to have a very low SQNR in comparison with the corresponding floating-point number format.

Apart from the efficiency of number representation, the cost of using the numbers also plays an important role. From the discussion in Section 2.1.3, it can be seen that; although multiplication of floating-point numbers is quite straight forward, addition of two floating point numbers is costlier than addition of two fixed-point numbers.

Custom Fixed-Point and Floating-Point Representations

In case of software implementation, the word-lengths of both fixed-point and floating-point representations are fixed or have limited choice. However, the actual format of the fixed-point representation is flexible to the extent that the binary point is not fixed. On typical software platforms, the fixed-point word-length sizes are generally a byte long (8 bits), half-word long (16 bits), one word (32 bits) and as long as double word (64 bits). The floating point numbers usually come in two formats: either as single precision or as double precision numbers. Further, it has to be noted that many processor platforms support SIMD[1] vector data operations. Using this, the given data path can be configured into desired number of bits in multiples of quads (4 bits) or bytes. The idea behind soft-SIMD [99] is to allow arbitrary word-length size by standard sized arithmetic data-paths to obtain finer control over fixed-point word-lengths.

The hardware implementation also opens up avenues for creating customized floating-point units [56; 124]. In [38], the floating point operator specifications are described using a parameterizable C++ floating point operators library. This is followed by automatic generation of optimal floating point operator implementations in hardware such that the time taken for computation is small enough to meet the desired frequency of operation. The impact of the total number of bits assigned to the floating point operator on the precision and dynamic range of the operation is not as straight forward as it is in the case of fixed-point number system. Under these circumstances, it becomes difficult to make a choice between the fixed-point and the floating-point formats without explicitly exploring all the options.

Some frameworks such as [6], are based on simulation and they eventually help

[1]Single Instruction Multiple Data

make a choice between the fixed-point and floating-point formats. It provides a parameterizable library to study the various trade-offs by way of simulation. In [45; 110] the dynamic range of the signal is considered to make a decision on the number representation. The choice of fixed-point number representation works well when the dynamic range of the computation is not large. Also, it is important that the required precision is small enough not to overshoot the cost of floating point costs. Many signal processing applications are linear and such systems can make use of scaling and can therefore work with normalized dynamic range (say, in the range $[-1, 1)$[1]). Also, signal processing applications and especially communication algorithms are designed to work in the presence of channel noise whose magnitude is much larger when compared to the quantization noise introduced even when 16 bit quantization is used. Therefore, the precision required for implementing these algorithms is also not very strict. In such circumstances it is possible to profitably use the fixed-point numbers effectively. *Therefore, communication and signal processing applications is the chosen area of focus in this thesis.*

Some works such as [52], make use of a hybrid number representation system. Here, instead of using a floating-point adder tree, the numbers to be added are normalized to the largest exponent and fixed-point addition is used to add the mantissa. This reduces the effort of computing the exponents every time. Another idea is to use the dual fixed-point number representation [41]. In this case, the exponent is reduced to a single bit and the mantissa consists of a signed fixed-point number. This is again a trade-off between dynamic range and precision by extending the concept of floating-point number representation. A third alternative is the use of block floating point number system. In this case, while a group of numbers share the exponent, the mantissa of such numbers can be different. Several signal processing algorithms have been implemented using block floating point numbers [61; 100]. Interestingly, they have found popular use in high dynamic range computer graphics applications. The *DirectX* library supports such a number format. The idea which is common to all these hybrid approaches is to be able to obtain the advantage of large dynamic range and high precision of the floating point representation while retaining the flexibility of the fixed-point representation.

2.2 Dynamic Range Estimation

The dynamic range of the signal affects the number of bits in the integer part of the fixed-point number. The range of any intermediary signal within a system depends on both the system function and the input signal distribution. In general, the signal can be multi-modal, asymmetric with non-zero mean. The dynamic range of the input signals is usually known by observing the representative test cases. In [91], the dynamic range interval is propagated from the input through every operator present in the system to the output using interval arithmetic (IA). This method is known to come up with conservative estimates and it does not take into consideration correlation between

[1] Right-half open set: A set $[a, b)$ defined on $\mathbb{R}$ is a set of real valued numbers consisting of all the numbers between points a and b on the real number line including a but excluding b.

various signals. In [58] it is proposed to use the *L1*-norm of the system transfer function to estimate the dynamic range. The dynamic range estimates obtained by using both IA and the *L1*-norm is equivalent for non-recursive LTI systems. The use of the *L1* norm is one of the most popular techniques for analytical determination of the dynamic range. Another technique which compensates for the inadequacies of the interval arithmetic is based on affine arithmetic (AA) as proposed in [29; 42]. This technique essentially keeps track of the first order correlation between various signals and is known to work well with linear systems. This would not work well when the signal distributions have asymmetric variations and when the variables have non-affine relations such as the relationship between the variables.

One drawback of IA and AA based technqiues is that it captures only the linear dependencies between signals. Application of these techniques on non-linear systems is known to provide very poor interval bounds. In [19], this problem is addressed by sufficiently splitting the input interval into many sub-intervals. The objective in this work is to be able to estimate the probability density function (PDF) of the signal distribtuion for each signal. When the interval contains a zero, the entire interval is split into several small intervals whose width decreases in geometric progression as the values approach zero. Split and Merge operations of PDF are formally defined for splitting the PDF and merging them after propagating the split intervals through systems.

It is straight forward to apply IA and AA based techniques to networks with feed-forward topologies. In [81], the AA method is extended for bounds analysis of systems with recursive topologies by adding an AA marker to each quantized signal. The given LTI system or a linear approximation of the non-linear system is considered in the state-space framework of modern control system theory and the actual bounds are calculated by propagating the input signal range by splitting it into multiple intervals. The classical IA is used to propagate individual split intervals. The actual bounds on any intermediatory signal is obtained by merging the results obtained by propagating individual input split interval.

The IA and AA techniques have been used more generically for purposes estimating numerical bounds on results obtained using computer arithmetic. The *GAPPA* tool [84] provides techniques for estimating IA and AA based bounds on polynomial expressions in order to validate and verify the code. This work can also used for fixed-point refinement purposes.

In [101], a technique based on the Karhunen-Loeve Expansion (KLE) is used to analytically determine the dynamic range of the signals in LTI systems. In [4], a technique based on KLE is described to study the propagation of PDF through LTI systems. This technique essentially propagates the characteristic function of the input signal distribution through the system function in the frequency domain to obtain the characteristic function of the output signal.

In [66; 68] the authors propose a simulation based technique to provide for a more accurate bound on the dynamic range. In their technique, signal statistics are acquired by performing a high-precision simulation of the system with representative inputs.

The first four statistical moments are extracted from the simulation data to calculate the skewness and the *Kurtosis* of the distribution of each signal. The signal density function (PDF[1]) is then characterized using these moments. The actual signal range is then estimated using the estimated PDF of the signal. The characterization of the PDF is as good as the input with which the system is tested. In general, it is required to conduct long simulation so as to characterize the inputs with enough diversity to represent the real world. In [102] the burden of simulation of all the input data is reduced by opting to perform a random simulation instead of exhaustive simulation. The authors exploit the fact that the maxima and minima statistics converge to the *Gumbel* distribution [69]. The simulation of the system using high precision arithmetic operators is carried out a number of times with different data but essentially similar statistical properties. This generates the required samples for constructing the Gumbel distributions for minimum and maximum value distributions. The range of the variable is then determined by considering the mean and standard deviation of the samples. This technique estimates the variance of the minimum and maximum values or the extreme values of the signal.

2.3 Accuracy Evaluation

The loss in accuracy is generally determined by computing a metric which can be obtained by comparing the output of the fixed-point implementation of the system with the output from a reference implementation. The reference design is implemented using double-precision floating-point numbers. The error induced into the computation due to double-precision arithmetic is very small. Thus, the double-precision implementation is often regarded to be as good as implementation with infinite precision arithmetic.

There can be several metrics that can quantify the error between the outputs obtained using fixed-point systems with respect to the reference outputs. For example, the bit error rate (BER) is a metric used to measure the performance of communication systems. The difference in BER can be considered to quantify the effect of quantization on the system output. More generally, the ratio between the signal power and the power of quantization noise, also referred to as signal to quantization noise ratio (SQNR) is used to measure the impact of fixed-point systems. In case of any signal processing system, often one would be interested in knowing the maximum error. In [70], the maximum quantization error is used as a metric to determine the impact of using fixed-point arithmetic. Designing filters is one of the most common tasks carried out for implementation of any signal processing system. In [31], the focus is on fixed-point filter design. Here, the actual error in the filter response due to quantization of the filter coefficients is used as a metric for measuring the impact of fixed-point arithmetic.

While using system specific metrics seems straight forward, it has several drawbacks. In case of a communication systems, a large sample set is required for simulation to make sure that the degradation in BER is not a random phenomena and it is indeed occurring due to the use of fixed-point arithmetic. This increases the time spent on

[1] Probability Density Function

BER simulation and is prohibitively long especially when performing word-length optimization which requires repeated fixed-point accuracy evaluation.

However, a more natural candidate for assessing the loss in accuracy is to measure the mean square error (MSE) between the quantized signal and the reference. This has been used in several earlier works such as [109]. The MSE can be expressed explicitly as a function of assigned fixed-point word-lengths making it easy to obtain analytical alternatives to simulation. Moreover, it is possible to use the MSE to derive SQNR and BER for a given system.

The MSE captures the total power in the error signal between the infinite precision (high precision floating-point) signal and its fixed-point counterpart. Therefore, MSE is also referred to as the quantization noise power [14]. A number of techniques to evaluate the quantization noise power due to fixed-point operations have been proposed and used in the literature.

2.3.1 Simulation Based Techniques

Using fixed-point simulation of systems under consideration, it is possible to determine the total quantization noise power due to the use of fixed-point numbers using the technique as depicted in Figure 2.4. The system is implemented using a reference design usually with very high precision. In case of input x, let y be the reference output. The loss in precision in such reference implementations is negligible and hence is approximated to be as good as infinite precision implementation. The system is also implemented using the proposed fixed-point formats and simulated. For every input x, let the corresponding output of the fixed-point system be $\grave{y}$. Then, the error e_y is obtained as a difference between the output of the fixed-point system with the reference output.

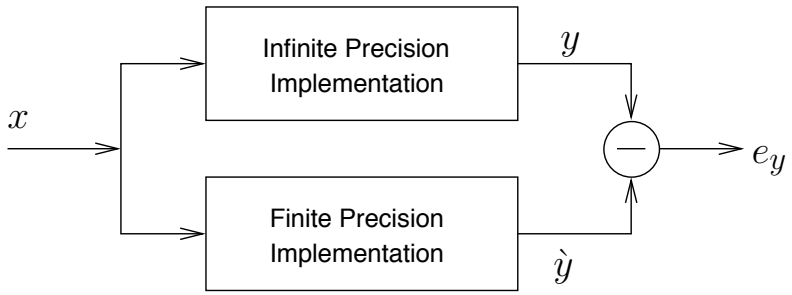


Figure 2.4: Measuring errors due to fixed-point

Several system-level design tools such as SPW [35], System-Studio [63], Matlab-Simulink [82] and Ptolemy [39] provide with fixed-point libraries for performing system-level simulation in their respective design environments. In [66], *gFix*; a parameterizable fixed-point data-type is proposed. It is also possible to use the *sc_fixed* data type in the *SystemC* environment to conduct bit-true simulation of the hardware models. The ability to perform bit-true simulations by these libraries is made available by means

of operator overloading. Operator overloading helps capturing the behavior of the fixed-point operator including the mode of quantization, the number of bits assigned and also effects such as saturation and overflow. Though using overloaded operators reduces the effort for setting up the fixed-point design prototype for simulation, the use of such non-native operations make fixed-point simulation costlier than the native floating-point simulation [64].

In order to improve the time for fixed-point simulation, in [66], the double precision floating point data type is used more efficiently. This technique makes use of the mantissa part only for all the fixed-point simulation needs. This technique restricts the number of bits assigned to a fixed-point number to 53. In [7], a similar technique is used for acceleration of the *SystemC* fixed-point libraries. The performance of fixed-point data types can be accelerated by using a suitable processor architecture (such as a DSP[1]). Techniques proposed in [37; 64] that were primarily designed for automatic fixed-point code generation can be used to target a given fixed-point format assignment on DSP platforms for quick execution. Another alternative is to make use of FPGA[2]s for hardware acceleration. The System Generator Tool from Xilinx [123] provides a suite of customizable fixed-point hardware modules commonly used for signal processing. These modules can be used along with Matlab-Simulink for quick prototyping of algorithms on hardware.

Simulation-based approaches are universal and can be applied without any restriction on any kind of system. However, the time taken for simulation [37] and the challenge of generating test vectors representative of the real life scenarios is a challenge. Nevertheless, ability to perform fixed-point simulations comes in very handy when there are no other alternatives or to make a first cut estimates of the impact of fixed-point systems. Several efforts based on simulation have been made in the past to measure the degradation in accuracy due to the use of fixed-point numbers.

2.3.2 Analytical Accuracy Evaluation

Measuring the errors due to quantization numerically by simulation is easy, but time consuming. Analytical techniques on the other hand, provide closed form expressions for calculating the quantization error statistics. As shown in Figure 2.5, the objective is to analytically determine b_y such that the output $\hat{y}$ is statistically equivalent to $\grave{y}$ that could be obtained by fixed-point simulation. This technique essentially aims to analytically calculate the statistical properties of e_y and model the additive component b_y suitably such that $\hat{y}$ is statistically equivalent to $\grave{y}$.

Such analytical techniques are a result of application of two theories: i) the analytical quantization noise model referred to as the pseudo-quantization-noise (PQN) model and ii) a noise propagation model based on the application of *"perturbation theory"*. The PQN model is based on the study of quantization error statistics and it explains the statistical properties of errors due to quantization at the output of a

[1]Digital Signal Processors
[2]Field Programmable Gate Array

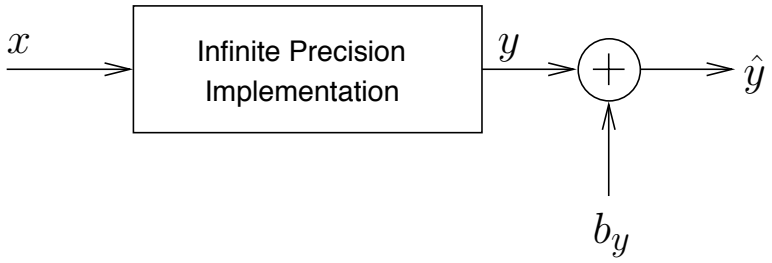


Figure 2.5: Analytically estimating error due to fixed-point operations

fixed-point quantizer. The linear noise propagation model is used to determine the impact of computation with finite precision inputs.

It has to be noted that, the efforts to estimate the statistical properties of the error due to fixed-point implementation with the reference implementation: e_y is restricted to knowing its quantization noise power. That is, to analytically obtain its first and second moments only. In this thesis, it is extended to include other statistical parameters such that it can be mimicked in greater detail.

PQN Model

The PQN model is also referred to as Widrow's quantization noise model [120]. It is a stochastic model that defines a random process which is statistically equivalent to the phenomena of uniform quantization. In other words, using the PQN model; the effect of uniform quantization can be modeled with an additive noise b to obtain $\hat{x}$ which is statistically equivalent to the signal $\grave{x}$ obtained after uniformly quantizing the signal x as shown in Figure 2.6. The PQN model further imposes conditions under which such statistical equivalence between the statistics of the noise signal b_x and the error e_x, the signals $\hat{x}$ and $\grave{x}$ can be established.

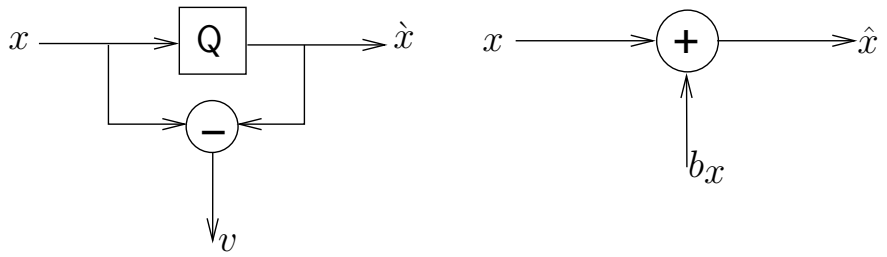


Figure 2.6: Fixed-point simulation vs. analytical quantization noise model

The ability to pre-judge signal statistics of e_x: which is dependent on the input signal and the quantization step-size is a key contribution of the PQN model. The characteristics of the random variable defined by the PQN model are defined as follows:

- Probability Density Function (PDF): The error signal b_x is uniformly distributed

and its range is determined by the quantization step-size. Mean of the random variable depends on the quantization mode (truncation, round-off).

- Additive Noise: The error signal b is statistically uncorrelated with the quantized signal. This means, the statistics of the quantized signal can be obtained by simply adding the statistics of the pseudo quantization noise b_x to the original signal statistics.

- Power Spectral Density: The power spectral density of the error signal b_x is white and the power is determined by the quantization step-size and the quantization mode.

The three statistical conditions are met when the quantization step-size is small in comparison to the dynamic range of the signal. More precisely, it is necessary that the characteristic function of the PDF of the signal is small enough such that there is no aliasing of the characteristic functions of the corresponding quantized signal. In such a situation, the process of quantization is referred to as smooth-quantization. Usage of the term *smooth* is to emphasise the fact that change in the actual quantized signal value is very small and only causes a proportionally small change in the system output. The actual conditions for each of these properties to hold are described by quantization theorems in [120].

Quantization Noise Sources

The quantization noise due to quantization of a real valued signal is given readily by the PQN model, the noise contributions due to coarser quantization of signals already quantized also follow the dynamics of quantization theorem. The quantization power is calculated as a function of the first and second order moments of the quantization error signal. The magnitude of quantization noise injected into the system by various fixed-point operations is determined by the fixed-point formats and the mode of quantization. No quantization noise is injected by an operation if the fixed-point format is sufficiently large.

Signals are subjected to quantization repeatedly in a fixed-point processing system. The mean and variance of the error due to the second quantization of the signal has also been studied. When the dynamic range of the signal is sufficiently large, a practical assumption that least significant bits assume a uniform distribution irrespective of the macro signal statistics is made [120]. The first two moments of the quantization error signal due to elimination of precision bits under such an assumption is as given in Table 2.1. Here, the column *continuous* refers to the effects of quantization when a real (or continuous in amplitude) signal is quantized using the weight of the LSB of the fixed-point format. The column *discrete* refers to effects of second and subsequent quantizations where k bits are eliminated from the fixed-point format with $(n+k)$ bits in the precision. The case of *discrete* case is that of the second quantization where a signal that is already quantized (hence discrete in its amplitude) is further quantized with larger step-size (lesser number of bits).

Fixed-Point Mode	Mean		Variance	
	discrete	continuous	discrete	continuous
Truncation	$\frac{q}{2}\left(1-2^{-k}\right)$	$\frac{q}{2}$	$\frac{q^2}{12}\left(1-2^{-2k}\right)$	$\frac{q^2}{12}$
Rounding	$\frac{q}{2}\left(2^{-k}\right)$	0	$\frac{q^2}{12}\left(1-2^{-2k}\right)$	$\frac{q^2}{12}$
Convergent Rounding	0	0	$\frac{q^2}{12}\left(1-2^{-2k}\right)$	$\frac{q^2}{12}$

Table 2.1: Mean and variance of quantization error

In the process of fixed-point refinement, the exercise of identifying noise sources has to be repeated before evaluation of output accuracy for any change in the assignment of fixed-point format

Noise Propagation

The small change in the output of the system due to fixed-point computations does not affect the overall macroscopic behavior of the system. The application of perturbation theory is essentially an attempt to approximately estimate the impact of quantization under such circumstances. The error due to computation with fixed-point operators are nothing but small perturbations whose effect on the output can be accounted for with a good degree of accuracy by linear approximation. Indeed, this approximation introduces a sense of tolerance towards quantization errors for an application and raises the question of how small should the quantization step be to be classified as smooth. An attempt to answer this question is made later in Chapter 4. Mentioned otherwise, it may be assumed that the quantizers are smooth. Also, the operator itself can be classified as smooth or un-smooth depending upon its functional behaviour. An operation is considered to be smooth if the output is a continuous and differentiable function of its inputs. The noise propagation technique based on perturbation theory, as presented in this section is applicable to systems consisting of smooth operators and smooth quantizers only.

Consider a binary operator whose inputs are x and y and the output is z. If the input signals are perturbed by b_x and b_y to obtain $\hat{x}$ and $\hat{y}$ respectively, the output is perturbed by the quantity b_z to obtain $\hat{z}$. In other words, as long as the fixed-point operator is smooth, the impact of small perturbations at the input translates to perturbation at the output of the operator without any change in its macroscopic behavior. In the realm of perturbation theory, the output noise b_z is a linear combination of the two input noises b_x and b_y:

$$b_z = \nu_1 b_x + \nu_2 b_y. \tag{2.13}$$

The underlying assumption here is that the noise terms b_x and b_y are uncorrelated with one another. This assumption follows from a more fundamental assumption made in the PQN model about the generation of quantization noise: that the quantization

noise is independent of the signal. The terms ν_1 and ν_2 are obtained from a first-order Taylor approximation [30] of the continuous and differentiable function f:

$$\overline{z} = f(\overline{x}, \overline{y}) \simeq f(x, y) + \frac{\partial f}{\partial x}(x, y).(\overline{x} - x) + \frac{\partial f}{\partial y}(x, y).(\overline{y} - y). \tag{2.14}$$

Thus, the expression of the terms ν_1 and ν_2 are given as

$$\nu_1 = \frac{\partial f}{\partial x}(x, y) \qquad \nu_2 = \frac{\partial f}{\partial y}(x, y). \tag{2.15}$$

It has to be noted here that the terms ν_1 and ν_2 can be time varying. This method is not limited to binary operations only. In fact, this method can be applied at the functional level with any number of inputs and outputs. This method can be applied on all operators on a given data path in order to propagate the quantization noise from its source to the output.

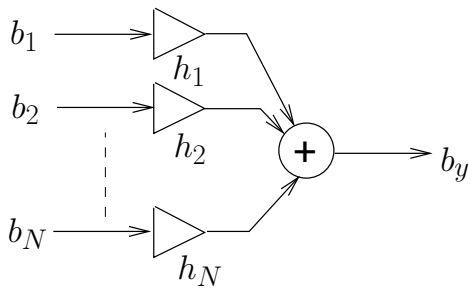


Figure 2.7: Propagating quantization noise power to system output

If the output of a system is denoted as signal y, the corresponding noise b_y due to quantization is obtained as the sum of contribution by individual noise sources in the system as shown in Figure 2.7. If $h_i(n)$ happens to be impulse response of the path from the i^{th} noise source to the output of the system, the contribution by the noise source b_i is computed as the convolution sum of the noise source and the impulse response. If the system consists of N noise sources and μ_{b_i}, $\sigma^2_{b_i}$ are respectively the mean and the variance of the i^{th} noise source b_i, there are N independent paths from each individual noise source to the system output. For each noise source b_i, a time varying impulse response h_i from the noise source to the system output is computed [87]. Its contribution to the output quantization noise b_y is obtained as the convolution sum with the associated time varying impulse response h_i. The total noise due to quantization at the output of the system is given as

$$b_y = \sum_{i=1}^{N} \sum_{k=-\infty}^{\infty} h_i(k, n) b_i(n - k), \tag{2.16}$$

where $h_i(k, n)$ is the value of the k^{th} time varying impulse response coefficient at the

n^{th} instant. The output noise power is obtained as the second order moment of the noise at the output of the system and written as:

$$\begin{aligned} \mathrm{E}[b_y^2] &= \mathrm{E}\Big[\sum_{i=1}^{N_e}\sum_{k=-\infty}^{\infty} h_i(k,n)b_i(n-k)\Big]^2 \\ &= \sum_{i=1}^{N_e}\sum_{k=-\infty}^{\infty} \mathrm{E}[h_i^2(k,n)]E[b_i(n-k)^2] \\ &\quad + \sum_{i=1}^{N_e}\sum_{j=1,j\neq i}^{N_e}\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} \mathrm{E}[h_i(k,n)h_j(l,n)]E[b_i(n-k)b_j(n-l)]. \end{aligned} \tag{2.17}$$

This can also be written as a weighted sum of the statistical parameters of the noise source as

$$\mathrm{E}[b_y^2] = \sum_{i=1}^{N_e} K_i\sigma_{b_i}^2 + \sum_{i=1}^{N_e}\sum_{j=1}^{N_e} L_{ij}\mu_{b_i}\mu_{bj}. \tag{2.18}$$

The terms K_i and L_{ij} are constants and depend on the path function h_i. This approach supports recursive/non-recursive systems in case of both LTI (linear time invariant) and non-LTI systems. The expression of these coefficients is given as

$$K_i = \sum_{k=-\infty}^{\infty} \mathrm{E}\left[h_i^2(k)\right]. \tag{2.19}$$

$$L_{ij} = \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} \mathrm{E}\left[h_i(k)h_j(l)\right]. \tag{2.20}$$

In [88], a method to determine the various path functions from the signal flow graph of the system is proposed. The given SFG is partitioned into many acyclic subgraphs for the purpose of analysis. Using the noise propagation model, the path functions of these sub graphs are determined in the *Z-transform* domain. The individual path functions from every noise source to the output is got by combining these sub-graph path functions by variable substitution.

An affine arithmetic based technique is proposed in [79] and [13] respectively for LTI and non-LTI systems. An affine form for every noise source in the system is defined by taking into consideration its mean and variance. Taking advantage of the linear noise-propagation model, the affine noise forms are propagated through the system by simulation. The values of K_i and L_{ij} are obtained by observing the affine forms of the output noise expression. The propagation is performed by a simulation technique. In case of a recursive system, this has to be repeated a number of times until the affine forms of the output noise value converges.

In [108], an analytical expression for the output quantization noise is derived using a vector format. This technique provides a uniform framework which includes non-linear, time-varying and recursive systems for estimation of output noise power. The analytical expressions are determined by performing a single floating point (or high precision) simulation and the knowledge of the respective path functions obtained by technique presented in [87]. In case of time varying systems, these expressions take the ensemble average of the time varying coefficients over the entire length of simulation. In case of recursive systems, the expression considers a large number of coefficients in order to approximate the infinite impulse response nature to obtain a pseudo impulse response character to the path functions.

In [112] the coefficients of the expression for output noise-power in Equation 2.18 are obtained by the second order Taylor-series expansion of the difference between the fixed-precision and infinite precision expressions. The Taylor coefficients essentially represent the K_i and L_{ij} coefficients considered by other works. The Taylor coefficients are obtained by solving a system of linear equations by performing minimum number of fixed-point and single floating point simulations. This is a one-time effort and once the coefficients are known, the evaluation of output quantization noise power is as trivial as evaluating the expression similar to the one in Equation 2.18

Apart from evaluating the exact expressions for the output quantization noise, attempts have been made to identify the bounds on the total output quantization noise power. In [121], the FRIDGE tool proposes an interval arithmetic based estimation of errors. Affine arithmetic based techniques have also been utilized to provide tighter bounds in [71; 73].

Some of the IA and AA based methods such as [19; 81] can be used for estimation of signal bounds analysis of quantized signals also. In [9] a polynomial expression to represent the bounds is derived and represented using *Handleman* representations. Heuristics are defined to estimate the maximum and minimum values of such representations to derive tighter bounds on signals. This method outperforms the previous IA and AA based methods for estimation of quantized signal bounds analysis for polynomial evaluations using fixed-point arithmetic.

2.3.3 Other Quantization Effects

Apart from contributing to the quantization error due to fixed-point operations, quantization of system coefficients indeed change the system behavior. Such changes are particularly important as they can affect the stability of the system under consideration. These effects are studied essentially under two categories.

The first among them is the impact of quantization of system parameters also referred to as coefficient quantization. In [80], the effect of coefficient quantization on the interval bounds of signal at the output is determined. It is clearly and rounding are clearly established that the coefficient quantization has a larger impact than the data quantization effects. This paper uses Interval arithmetic based techniques. In [55; 75], this effect is studied in the z-transform and Fourier transform domain by observing the sensitivity of each coefficient with respect to the system transfer function. A simple

simulation based technique to calculate the sensitivity is proposed in [77].

The second effect is due to the non-linearity of the quantization process in the presence of recursive loops. Such effects under certain conditions such as the ones discussed in [25; 90] can cause oscillations. Techniques to detect the existence of limit cycles due to quantization have been proposed in [5; 62; 107]. In these techniques, the number of bits assigned to operators is limited to a small finite number. The analytical techniques focus on deriving sufficient conditions for fixed-point quantization such that limit cycles do not occur [118]. Some other analytical techniques such as [10; 11; 89] aim at deriving conditions for guaranteeing asymptotic stability of the systems. These analytical techniques are supported by simulation when it is too complex to derive analytical expressions. In [78], affine arithmetic based techniques have been used to accelerate simulation. The simulation based techniques require exhaustive simulation and therefore can be very time consuming [16].

2.4 Cost Evaluation

The problem of estimating the cost for a given choice of fixed-point word-length is the other very significant part of the word-length optimization process. After all, it is the cost which needs to be minimized in order to achieve optimality. Referring back to Figure 1.1 in Chapter 1, if the estimated cost is not equal to the actual cost, it can be known only after the system is implemented. Therefore the cost models used for estimating the impact on system cost due fixed-point word-length choices has to be as accurate and reliable as possible.

In general, there are a number of NP-complete problems such as scheduling and resource binding that need to be solved soon after the fixed-point refinement step. It is therefore difficult to exactly measure the cost. Drawing from the design techniques, a cost model which reflects the trend in the cost evolution has to be built. Since, this is a very early attempt to estimate costs in the design process, these are referred to as high-level cost estimates. These estimates are typically that of area, power and execution time of the implementation.

In [97], an attempt to derive a generic hardware cost model is made. Here, the area of each hardware block is parameterized by the fixed-point word-length and stored in a library. The total area of an implementation which uses the operators of various word-lengths is obtained by individually looking up this library and adding up the individual operator costs. Similarly, the library can be parameterize by considering other cost metrics such as power consumption, energy dissipation, power-delay product etc. as a function of operator word-lengths [94]. In [111], the cost is modeled as a quadratic function of fixed-point word-lengths. They use a hardware based resource estimator to obtain the actual implementation cost for some samples in the word-length space. A least-square problem is formulated from the data obtained by using the estimator and the coefficients are obtained by solving the least-square problem.

In [15], a comprehensive cost metric which takes into account the area of fixed-point arithmetic operator, area of glue logic and the area of registers on FPGA platforms is

defined. The metric is described as a combination of the infinite norm and the first norm of the area metric. Although the comprehensive nature of such a metric is helpful in capturing the actual cost, this technique essentially tries to solve the word-length optimization problem by taking into consideration resource area estimates by using an approximate list-based scheduling algorithm. Given the NP-hard nature of the word-length optimization problem in itself, these techniques donot scale well for use in system level design.

2.5 Word-Length Optimization

The cost and accuracy of systems implemented using fixed-point arithmetic are functions of the assigned word-lengths. Given that there are a number of options for choosing the word-length size, the choice of fixed-point formats for each of the signals has to be done carefully.

For discussion in this section, let $\mathbf{w}$ be a vector of word-lengths with each entry corresponding to exactly one operation in the fixed-point implementation. Let the closed form expression for total cost evaluation function of the fixed-point implementation and its accuracy quantifying metric evaluation function be given as $C(\mathbf{w})$ and $\lambda(\mathbf{w})$ respectively.

2.5.1 Problem Variants

The trade-off between implementation cost and the accuracy of computation is essentially made either to achieve the best performance given the constraints on the cost or to achieve low cost design given that the performance criteria is constrained by a minimum bound. Consequently, there can be two variants of the word-length optimization problem.

Performance Maximization Problem In this problem, the cost of implementation is the constraint. Given the maximum cost budget C_{budget}, the objective is to maximize the system accuracy λ. This problem can be formally stated as

$$\max\left(\lambda\left(\mathbf{w}\right)\right) \quad \text{subject} \quad \text{to} \quad C\left(\mathbf{w}\right) \leq C_{\text{budget}}. \tag{2.21}$$

Cost Minimization Problem In this problem, the performance is constrained instead of the cost. Given the maximum permissible quantization noise $\lambda_{\text{objective}}$, the objective of the optimization problem is to minimize the cost C of its implementation. This is formally stated as

$$\min\left(C\left(\mathbf{w}\right)\right) \quad \text{subject} \quad \text{to} \quad \lambda\left(\mathbf{w}\right) \leq \lambda_{\text{objective}}. \tag{2.22}$$

Selecting between the two versions of optimization problems is driven by design requirements. The designer may want to choose to solve the *Cost Minimization* problem if there are strict accuracy constraints while allowing enough margin for the total cost.

On the other hand, designers choice could be the *Performance Maximization* if there are strict constraints on the cost of implementation while compromising on the accuracy of computation. In an ideal situation, it is expected that the trade-off curve obtained by solving either problem for various target constraints are one and the same. This corresponds to the Pareto-optimal front of the given system. As the solution to either optimization problems are driven by heuristics, this may not be the behavior in practice.

Optimization Variables

A signal processing system consists of hundreds of fixed-point operations. For example, a 64-point FFT signal processing algorithm has 960 fixed-point operations including addition and multiplication with constants. The integer and the fractional parts of fixed-point format is chosen such that it accommodates the dynamic range of all signals and has enough precision. On a flexible architecture such as an FPGA or while designing an ASIC or the modern day DSP architectures with SIMD support, it is possible to have flexible data widths. To solve the word-length optmization problem, each of the choice of word-lengths of each of the fixed-point operations has to be assigned optimally.

Let M be the number of signals in the system. If each of these signals can be assigned N different word-length configurations, there can be N^M different word-length vector combinations. This is referred to as the multiple word-length assignment (MWL) paradigm for fixed-point refinement. The complexity of evaluating the cost-performance trade-off for each of these combinations is practically impossible.

Assigning the same word-length format for each signal reduces the complexity of the word-length optimization problem. This is the popular uniform word-length assignment (UWL) paradigm for fixed-point refinement. In this approach, all the computations are carried out with the same word-length. In other words, all the signals and operators in the system are assigned the same fixed-point format. This approach for fixed-point design of signal processing systems was predominantly used for implementation on fixed-width data-path architectures. Then, the integer and fractional parts of the fixed-point format is chosen such that the dynamic range of all signals are accommodated and there is enough precision for all signals. If the uniform word-length paradigm were to be used, the number of word-length combinations that are tried before reaching optimality is N, where N is the number of word-length configurations realizable on the given architecture.

2.5.2 Solutions to the Word-length Optimization Problem

In the past, the fixed-point refinement problem was addressed using the uniform word-length (UWL) paradigm. In the UWL paradigm, all the signals are assigned to the same fixed-point word-length and the same format. On the contrary, each signal is assigned a suitable fixed-point format depending on its sensitivity to the system output. It has been shown that the (multiple word-length) MWL paradigm often provides better results than the fixed-point formats obtained by using the UWL paradigm [31]. The benefits of solving the fixed-point refinement problem using the MWL paradigm comes

at the cost of solving an NP-hard optimization problem. Whereas the fixed-point refinement solutions is simply $O(N)$, where N is the various choice of word-length formats if the UWL paradigm is used instead.

Various techniques to solve the word-length optimization problems can be categorized into optimal, stochastic and quick-heuristic based approaches.

Optimal Approach

Attempting to solve the word-length optimization in the true sense of optimality requires exploration of the entire combinatorial search space before the optimal solution is got. In [50], an attempt is made to narrow down the search space by estimating the minimum and maximum bounds on the variables. In order to calculate the maximum bound (MaxWL), the word-lengths of all operations are uniformly increased until the accuracy criteria is met as described in [23]. In [115], the minimum bound (MinWL) is calculated by iteratively decrementing the number of bits assigned to each variable. In this process, the variables are uniformly assigned a maximum number of bits. Typically, word size (32 bits) or double word sizes (64 bits) are assigned or double precision operations are used. Then, one variable is considered at a time while the other variables continue to be either assigned the initial number of bits or assigned double precision floating point format. The number of bits assigned to the chosen variable in any given time is decremented until the performance becomes inferior to the specified target computational performance.

The word-length optimization problem is formulated as a mixed integer linear programming (MILP) problem for LTI[1] systems in [31]. The noise power expression as a function of assigned bits is exponential in nature. In order to linearize this in the context of the proposed MILP formulation, as many auxiliary variables as the number of bits permissible in the constrained search space is introduced. So, if there are M variables with N different word-length choices, there would be $N \times M$ number of MILP variables. Having such large number of MILP variables can take a long time. A number of constraints depending upon the implicit dependencies between the signals is introduced such that the search space of the MILP problem is further reduced. These constraints are obtained by performing basic checks including word-length propagation and the constraints imposed by the cost model. Owing to the number of variables and the fact that such huge number of constraints have to be dealt with, this technique clearly does not scale very well with large systems.

Moreover, it is known that the multiple variable word-length optimization problem is non-convex [27] and NP-hard [34]. Therefore, searching in limited search space many not be really effective and it cannot guarantee that the reduced search space has not excluded the optimal solution vector.

[1] Linear Time Invariant

Quick-heuristics Approach

Given the NP-hard nature of the word-length optimization problem, a number of heuristics have been proposed as a solution to this problem. The computation of the minimum and maximum bounds on the number of bits that can be assigned to the operators stands out commonly amongst many approaches. In [17], a number of word-length optimization heuristics based on different ways of calculating the minimum and maximum word-lengths are summarized and compared against one another.

Some of these heuristics begin with the determination of minimum word-length (MinWL). The bit-widths of all the signals are incremented iteratively until the specified target computational performance is not met. In every iteration, the number of bits assigned to one of the fixed-point signal variables is increased by b bits. The choice of this variable is driven by the cost, accuracy trade-off. This technique is common to a class of algorithms referred to as *min +b bits* algorithm.

While [31] uses the criteria of minimum cost increment, the maximum improvement in accuracy is used in [18]. In [50], a heuristic which takes a weighted average of the loss in precision and the increase in the cost is used for determining the direction of optimization in each step. In general, using the cost alone is known to take a large number of iterations whereas, the use of just accuracy requires lesser number of iterations but is complex due to the fixed-point performance evaluation involved in every iteration. Using the meta heuristic, the influence of cost factor and the accuracy factor is weighted such that the number of iterations are minimized at the same time not compromising on the quality of the solution. In [17; 51], the ratio of the cost derivative and the accuracy derivative with respect to the changing number of bits is considered. The signal which gives the best improvement in the accuracy for the smallest increment in cost is considered.

On the contrary, it is also possible to begin with the maximum bit-width assignment [31] (MaxWL) and decrement one of the bits assigned to the signals iteratively. The criteria to choose the signal whose bit-width has to be decremented takes into account the accuracy and cost trade-off is similar to the criteria discussed for algorithms that increment the bits assigned iteratively. This is referred to as the *max -1 bit* algorithm.

In [116], a hybrid heuristic procedure which combines strategy to both increment and decrement the number of bits is discussed. After calculating the minimum word-length combination, all signals are uniformly incremented by one bit until the performance criteria is met. At this stage, the different signals compete to lose one bit iteratively such that the accuracy criteria is not violated while minimizing the total cost. In [96], selectively increments or decrements the bit-widths to arrive at an optimal solution. The gradient obtained as the ratio of the change in cost and the change in accuracy is considered to choose the fixed-point variable whose word-length has to be modified in each iteration. As long as the accuracy is inferior to the expected levels, the algorithm is in the increment mode. If the accuracy constraint is over shot, the direction is reversed and it is in the bit decrement mode. Only those variables which contribute in the right direction in either of the modes is considered. Other variables are

moved to the *taboo* list. Iterations are carried out until all the varaibles are eventually not added into the *taboo* list.

In [23], a branch and bound algorithm which computes the solution between the maximum and the minimum word-length combinations is proposed. This approach is also the basis for the optimization technique used in [86].

Stochastic Approach

The use of simulated annealing (SA) technique for word-length optimization can be found in the very early work [20]. In this approach, the word-length optimization problem is formulated as a MILP and then SA is used to arrive at the solution of the MILP problem. In simulated annealing, every change in the values assigned to the optimization variables is referred to as a movement. Every time a movement is performed, the error power and the cost of implementation is evaluated. All solutions obtained are preserved for comparison with solutions obtained by movements in future. If a movement generates a superior solution than the best available thus far in any iteration, it is taken with a probability of 1. Otherwise, a smaller probability is assigned to it depending upon the quality of the solution obtained. The various movements can be thought of generating different states in a Markov-chain. This probability decreases with every passing iteration. If a proper initial solution and a proper annealing function is chosen, the optimal solution for the given problem can be found quickly. In [72], an adaptive simulated annealing procedure is proposed along with the use of minimum word-length determined by the techniques discussed in the previous section. The authors caution against using this technique for very large designs as it can consume large amounts of time.

In [92], genetic algorithms (GA) have been adapted for word-length determination of the *LMS* filter. In any genetic algorithm, a string of genetic elements are assigned to each of the optimization variables. These elements contribute to the behavior of the variables in the system. This string of elements is often referred to as chromosomes. In every generation, a sub-string from the strings are randomly selected and exchanged between two possible solutions. The resulting solution is referred to as the child of the two random solutions. This constitutes the next generation solution. Once the solution is obtained, the survivability of the solution depends on its performance. A survivability criteria is determined by evaluating the cost and accuracy of the solution obtained. The candidates obtained by allowing many generations to breed by tuning the survivability criteria towards optimality, an optimal solution is obtained. A hybrid technique which uses both the GA approach along with a gradient based heuristic approach is explored in [95] leading to faster convergence than just using the GA approach. Another variant of the GA approach is the multi-objective GA which is useful in case of signal processing systems with many outputs. In this variant, the GA is guided such that multiple objectives as defined by the performance criteria for each system output is satisfied. In [114], this technique is used to determine the coefficients of the FFT butterfly in the context of an MC-CDMA receiver. Another example of using the multi-objective GA is found in [1]. Here, the DCT and the FIR filter coefficients jointly participate in

the evolution of the solution. In [49], it has been found that the GA approach works well and produces better solutions than the greedy approaches when the problem size is small.

In [96], a greedy randomized adaptive search procedure (GRASP) based algorithm is proposed. It is essentially a trade-off between the genetic algorithmic approach and the quick heuristic approaches. The algorithm begins with minimum word-length assignment to each of the fixed-point variable. A greedy randomized algorithm is used instead of a deterministic algorithm in every iteration to pick the next eligible candidate for word-length increments. While the heuristic algorithms usually decide upon one of the word-length optimization variables, this algorithm chooses one among equally eligible variables randomly instead. The randomization reduces the chances of getting stuck in a local minimum. When the iterations eventually converge onto the target quantization noise power, a local search using the *taboo search* algorithm around the solution thus obtained is performed using a deterministic heuristic. It is observed that when making the choice of bits assigned to each fixed-point variable, incremental word-length assignments tend to over optimize the system. That is, while the assigned fixed-point format does not meet the optimization criteria; the next iteration improves the fixed-point performance to an extent beyond what is expected. Therefore, the bit-widths are adaptively incremented or decremented as a part of this heuristic depending upon the placement of the iteration with respect to the randomized greedy solution obtained in every optimization step.

Analytical Approaches

In this approach, the attempt is to mathematically obtain the optimal solution for the given optimizaiton problem. A mathematical treatment to the steepest gradient descent approach is provided in [21; 44]. The authors proceed to propose their own version of the steepest descent algorithm which is similar to the ones proposed with minimum word-lengths but just that the minimum is set to 0 bit to the precision for all signals. Identifying the similarities between the word-length optimization problem and the bit-allocation problem in the signal compression domain, this technique is also referred to as *marginal analysis*. Both [21; 44] attempt to provide an analytical basis for the optimization procedure based on *Lagrange Multipliers*. This is achieved by relaxing the constraint of assigning integer values thereby making the cost and accuracy evaluation function convex.

One disadvantage with the *Lagrange Multiplier* based techinques is the possibility of having negative values for the word-lengths. The authors suggest that such results may be arrived at when optimizing for very large quantization noise magnitudes. In [22], the authors overcome the problem of negative bit-width assignment in [21] by transforming the word-length optimization problem into a *Geometric Programming* problem.

In [43], an objective minimization function similar to the distortion and sensitivity metric is used for performing the cost and performance trade-off. The paper assumes a quadratic cost model and discusses an iterative algorithm to arrive at the bit-widths of each participating variable. Sufficient conditions to ensure convergence of the iterations

onto optimality is also provided. This is superior to using the *Lagrange* multiplier based technique as there is no risk of ending up with negative bit-width assignments for the precision part and sufficient precautions such that the iterations converge is already taken into account.

In [27], the authors consider the non-linear effect of the noise generation as a function of the number of bits assigned. They identify this to be a problem where allowing freedom for all the word-length variables independently causes non-convexity. An algorithm based on sensitivity to quantization noise is used to constrain the word-length variables such that convexity of the word-length optimization problem is preserved.

2.6 Automatic Fixed-point Refinement

The accuracy evaluation and cost evaluation are just estimates of the system when eventually implemented. The actual values could be off the estimated marks. The inability to account for all the factors affecting the design implementation due to the steps that are carried out post fixed-point is the reason for such deviations. Then, the designer needs to repeat the entire design flow by suitably adjusting the target accuracy or target cost or both in order to obtain the designed result. In such circumstances, the various sub-tasks within the fixed-point refinement step may have to be performed repetitively until the system parameters converge onto the objectives set out by the designer. Therefore, development of automatic tools for performing fixed-point design has evoked a lot of interest among the research community. A number of automatic fixed-point refinement tools have been designed as a result of various academic research pursuits. This section focuses on the efforts for automation of the fixed-point refinement process.

The need for fixed-point refinement framework was felt very early. In [68], the authors propose a pseudo assembly instruction set which is similar to the then popular DSP platform TMS320C25 from Texas Instruments. The algorithm is first coded using the proposed pseudo instruction set. A simulator which interprets such a pseudo assembly program is provided as a part of this fixed-point refinement framework. The tool then obtains the dynamic range of each of the signals and determines the scaling required for each of the signals. Using this information, the fixed-point number formats are automatically determined and the native TMS320C25 instructions are automatically generated. This idea was extended to higher level languages in [67]. In this approach, the *fSig* class library designed to collect signal statistics is used for performing the dynamic range analysis. The collected statistics is used to determine a suitable fixed-point data type. The *gFix* library provides the necessary fixed-point operators for simulation of the fixed-point system. An automated tool using the SUIF [24] compiler framework is developed for conversion of any given floating point algorithm to work with fixed-point data types.

Another early attempt is the *FRIDGE*[1] framework [121]. This effort is based on the *Fixed-C* extension to the ISO-C programming language. This extension essentially in-

[1] Fixed-point pRogramming DesiGn Environment

troduces the fixed-point data type into the C programming environment. The dynamic range and the precision bits corresponding to every signal is explicitly annotated. Since none of the native C compilers provide support true fixed-point formats, the *Fixed-C* compiler [59] (FCC) is developed to provide compiler level support for the fixed-point data type introduced in this framework. A complete fixed-point specification of the program requires the fixed-point format of each signal to be defined explicitly. Doing this manually can be very cumbersome and time consuming. Therefore, this tool captures the non-exhaustive fixed-point specification provided by designer and suitably assigns fixed-point formats to the other signals too. To do this, the *HYBRIS* simulator which simulates both fixed-point and floating point arithmetic together is developed. The *ICEPACK* utility assists in identifying the impact of fixed-point constraints defined by the user on other signals whose fixed-point formats are not explicitly defined. This is followed by platform specific code generation which includes ANSI-C for DSP processors and behavioral VHDL for hardware development. This tool was adopted by Synopsys in their CoCentric suite used for hardware/software co-desigin and implementation.

Both approaches above are essentially based on simulation and needs user intervention in all the stages of fixed-point design except for setting up a fixed-point simulation platform. In [28] a quasi simulation approach where the dynamic range and the quantization error due to limited precision is propagated through the operator along with a high precision simulation. This is achieved by defining a new overloaded operator which takes care of both functionality and the finite word-length aspect. The fact that simulation in high precision and fixed-point accuracy is performed together provides an opportunity to dynamically monitor the range and keep track of the error in accuracy. The ability to simultaneously perform fixed-point refinement while performing simulation overcomes the need for maintaining huge databases of nominal signal values. It is possible to steer the evolution of the floating point simulation in the direction of the fixed-point algorithm at junctures where control decisions are to be taken. Thus, the decision errors induced due to fixed-point computations are already taken care of at the end of the simulation. Though this method is simple and quickly provides a way of realising any algorithm in the fixed-point format, this technique is oblivious of optimality concerns.

In [33], the Synoptix tool begins with a data-flow description of systems in Matlab's Simulink environment. The dynamic range of the signal is propagated to calculate appropriate scaling of the fixed-point formats either by using the L_1 norm or otherwise. Without going into the intricacies of the dynamic range estimation, the authors assume that the integer and fractional bits are thus specified. The authors assume truncation mode for all quantization operations for the sake of simplicity. They define one of the greedy heuristics *Max -1bit* for automatic word-length optimization. The fixed-point representation thus obtained is used by a HDL synthesis tool along with custom fixed-point operator libraries to realise the complete design in hardware. A similar fixed-point refinement tool based on Simulink is also presented in [111]. The word-length optimization problem is cast as a mixed-integer linear programming problem (MILP). In [93], an analytical accuracy evaluation technique which uses a forward error

propagation is used for fixed-point conversion of algorithms designed in MATLAB to generate hardware designs using VHDL.

In [85], the problem of source to source transformation of an algorithm expressed using floating point algorithms is addressed. The ID.Fix tool is implemented using the GECOS framework and contrary to Simulink descriptions, it works with C or C++ code with *#pragma* directives suggesting the dynamic range of the fixed-point numbers are taken as the input. The signal flow graph of the algorithm is extracted by inference [47]. A fully automatic analytical technique developed in [87; 108] are applied on the graph in order to generate the analytical expressions of the loss in accuracy due to finite precision at the output of the system. A simple cost metric is used to estimate the cost as a sum of individual operator cost. The modular nature of the tool and the GECOS platform allows experimentation with optimization heuristics. Many optimization heuristics have been proposed and used successfully in this context. The focus on the previous approaches not just realization of the system using fixed-point arithmetic and rather it is the optimality of the fixed-point solution.

2.7 System Level Approaches

The work in [119] describes a high level synthesis flow of a MIMO-OFDM algorithm design to hardware functioning in real time using Matlab and custom high level synthesis framework. The authors further propose a dependable and repeatable methodology to address all the problems in the process of realising an algorithm in hardware. In the first of its kind of attempt, the authors propose a design flow where the top-level fixed-point formats are first determined. In this approach, the choices for input and output fixed-point formats of the sub-systems such as the FFT block, filters and equalizers are first explored. Once this is established, the individual signal formats are determined in the next step while respecting the input output fixed-point formats of the sub-system. In this process, the total quantization noise is continuously monitored during every fixed-point refinement step. Given the multiple abstractions in which the fixed-point refinement is carried out, this can be regarded as the first true system level word-length optimization effort.

As it was mentioned in Chapter 1, it is clear that the word-length optimization problem is one among many other problems that need to be addressed in a typical high-level design flow for embedded systems or digital hardware design. Since the word-length optimization is performed very early in the high-level synthesis flow, there is no information about the actual behavior of the system when different combinations of the word-lengths are used. The cost function from the perspective of the word-length optimization problem is an estimator which looks ahead and predicts the total cost of the system.

Therefore, the cost function is used during word-length optimization is an over simplified model of the actual costs. In practice, though the simplified cost function complies with the macro behavioral model, the actual cost function is seldom as simple as it is made out to be. This is due to the impact of the choices made during subsequent

steps such as scheduling, binding which are NP-complete by definition can have a profound effect on the over all system costs. In other words, the high complexity of the later steps makes it impossible to accurately quantify the actual cost of implementation before actually implementing the given system.

Many attempts to jointly solve the word-length optimization problem along with scheduling and binding problems have been attempted [32]. The scheduling and binding stages are themselves NP complete problems and hence clearly the joint problem is also combinatorial in nature and is more complex than these problems considered individually.

Attempts such as [36] show the efficacy of the platforms which can make good use of fine grain word-length assignment in the context of of software defined radio is addressed. It is interesting as it shows that systems which make use of the multiple word-length fixed-point refinement process can be well exploited. This work focuses on fixed-point refinement for software defined radio (SDR) algorithms constrained by the total energy dissipation. In particular, the focus is on design of efficient ASIP[1] platforms for SDR algorithms.

This thesis addresses the problem of dividing the given problem at the algorithmic abstraction by dividing it into a hierarchy of three levels of abstraction. The system level or the global level encompasses the entire system under consideration. The functional partition level is set one level below the system hierarchy. These sub-systems can be thought of logical components used in building the system. There can be sub-systems defined at levels below the functional partition depending upon the complexity of optimization and accuracy evaluation. Here, the functionality of the partition is divided into many smooth clusters consisting of operations whose response to quantization noise behavior can be captured analytically. In the third level, are the actual fixed-point operations that eventually realise the sub-systems. The fixed-point refinement problem is essentially that of arriving at the optimal assignment of word-lengths. The main focus of this work is on designing custom data paths for ASIP in order to realise the algorithms refined in fixed-point effectively.

2.8 Summary

The use of fixed-point refinement has immense practical benefits especially for signal processing applications. Previous attempts to address the problem of automatic fixed-point refinement discussed in this chapter have several limitations. To begin with, the analytical techniques for accuracy evaluation are limited to certain operators classified as smooth. Therefore, it is inevitable to use simulation based techniques in the presence of operators that are not smooth. Secondly, the word-length optimization algorithms do not scale well with growing system sizes and the increasing number of variables participating in the optimization. Though the work in [36] addresses this problem by proposing a hierarchical divide-and-conquer technique, it is limited to performance evaluation only.

[1] Application Specific Instruction Processor

In this thesis, shortcomings of some of these techniques are addressed. The focus in this thesis is on the scalability of the word-length optimization technique and the associated accuracy evaluation problem. In the interest of time, other implementation issues such as coefficient quantization, limit cycles and the estimation of dynamic range of the signal are not addressed in this thesis. Also, the intricacies of a realistic cost estimation function are approximated with simple affine cost models.

Chapter 3

Single Noise Source Model

Each fixed-point operation in a given application can be a potential source of quantization noise in fixed-point system implementation. Using the analytical techniques based on PQN (Pseudo Quantization Noise) model that were discussed in the previous chapter, it is required to calculate the noise power at each of the operator noise sources and their respective path gains to arrive at an analytical expression for quantization noise at the system output. With growing system size, the number of operators and hence the number of noise sources and the respective paths to the output increase and quickly become very large in number. Consequently, even evaluation of quantization noise power at the output of the system takes longer with every new quantization noise source introduced into the system. Moreover, the number of optimization variables increase with the number of noise sources leading to increased complexity of word-length optimization as the evaluation of fixed-point precision accuracy has to be carried out more often.

In this chapter, the problem of estimating the error due to quantization at the output of a large system is considered. The idea of hierarchical decomposition of the given signal processing system is proposed to overcome the problem of long accuracy performance estimation time. Moreover, this way of decomposition of a system into subsystem hierarchies forms the basis of divide-and-conquer approach to solve the word-length optimization algorithm presented in chapter 5. From the hierarchical system view point, large system is broken down into a hierarchy of sub-systems arranged in multiple levels. The single-noise-source (SNS) model is conceptualized to extend the basic operator level noise-power estimation technique to the hierarchical entities describing the system.

The SNS model is essentially an attempt to model the errors due to quantization noise as a random process and to derive the relevant parameters which can mimic the random process without having to perform fixed-point simulation.

3.1 Hierarchically Defined Systems

The evaluation of accuracy at the output of a fixed-point system is one of the most important performance metric for the performance of a fixed-point system. The use of linear noise propagation models presents an opportunity to calculate the total quantization noise power at the output as a function of the quantization noise power at each of the operator sources. In the hierarchical approach, the system is decomposed into many smaller sub-systems. Thereby, all the operators and the corresponding noise sources are clustered into groups defined by the sub-system boundary.

3.1.1 Hierarchical Decomposition

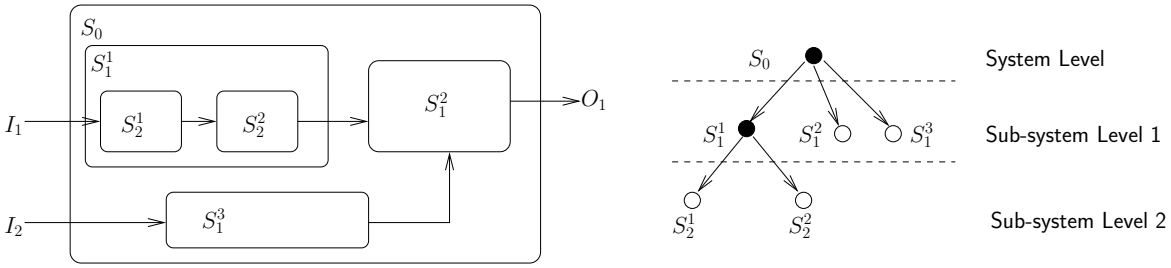


Figure 3.1: Hierarchical decomposition of the system S_0

Consider a hierarchically defined system as shown in Figure 3.1. The system S_0 has two inputs I_1, I_2 and an output O_1. The system is decomposed into two levels of hierarchies. At the system-level, S_0 is divided into sub-systems S_1^1, S_1^2 and S_1^3. In the next level, the sub-system S_1^1 is further sub-divided into S_2^1 and S_2^2. The hierarchical decomposition is also shown as a tree with nodes representing sub-systems at different hierarchical levels. Each node represents the system or the sub-system-level abstraction. The edges point to nodes in the direction of decomposition of the system size with lesser complexity. While the root node is only one and it represents the entire system under consideration, all nodes in each successive hierarchical level together represent the system. With every successive hierarchical level, the number of nodes increases and the complexity of each of the nodes reduces progressively relative to its parent node.

The smallest sub-systems that are not divided into sub-hierarchies any further are found at the leaf positions in this tree and are hence referred to as leaf-sub-system. In this example, the sub-systems S_2^1, S_2^2, S_1^2 and S_1^3 are leaf-sub-systems. Hierarchical decomposition of the system is primarily governed by the complexity of accuracy evaluation of the smallest sub-system. Theoretically, the division of the given system into sub-systems can be extended all the way down to the operator level. Dividing the system into many small sub-systems can sometimes be counter productive as the number of sub-systems that needs to be considered at the successive higher level of abstraction can be very large while the complexity of the leaf level sub-system can be very trivial (such is the case when the decomposition is performed to the level of an operator).

Therefore, a good balance between the complexity of each leaf-level sub-system and the number of sub-systems needs to be identified. Apart from the complexity concerns, there can be operators whose response to quantization noise input cannot be modeled analytically. Such types of operators is not made a part of any sub-system. While there is nothing wrong in principle to have any type of operator within a given sub-system, it is not possible to arrive at an analytical expression at the sub-system-level in the presence of operators whose response to quantization noise input cannot be modeled analytically. Thereby, fixed-point simulation will become necessary if such operators are included within the sub-system. Keeping this in mind, the sub-systems are defined such that it is possible to arrive at the parameters of the random process mimicking the quantization noise errors analytically and doing so using the techniques presented in Chapter 2 becomes manageable in terms of its complexity.

Signal Flow Graph (SFG) Representation

Signal processing systems can be graphically represented by using signal flow graphs (SFG). A signal flow graph defined at the operation level consists of various operators as nodes inter connected by directed edges pointing in the direction of signal flow. The SFG of a hierarchically decomposed system, essentially consists of its sub-systems. Depending upon the level of abstraction chosen to build the SFG, it is represented by the sub-systems defined at the chosen hierarchy interconnected with directed edges pointing in the direction of the flow of the signal. Such an SFG has lesser nodes when compared to the SFG defined using the basic operations. From Widrow's quantization noise theory, it is clear that the quantization noise can be considered to be statistically uncorrelated with the signal and that it is additive in nature. Therefore, the fixed-point system can be represented by augmenting the SFG with quantization noise sources. Such a noise source represents the total quantization noise generated due to fixed-point operations within the given sub-system. This noise source is referred to as the *single-noise-source* (SNS). Modeling the characteristics of such noise sources is discussed in the major portion of this chapter starting from Section 3.2.

The *Augmented-SFG* graph representing the fixed-point system is constructed from the system SFG by adding a quantization noise source corresponding to each sub-system. The noise generated within each of the sub-systems is added at the output of the respective sub-system. Thus, there are as many quantization noise sources as the number of sub-systems defined in hierarchy at a given level of abstraction. In case of the example considered in Figure 3.1, the *Augmented-SFG* representing the fixed-point implementation of the system is shown in the Figure 3.2. The *Augmented-SFG* is obtained by adding the noise generated from within the leaf level sub-systems, it corresponds to the lowest or the finest hierarchical definition. The quantization noise sources at higher levels of abstraction can be derived as a function of the leaf level sub-system output quantization noise. Deriving the noise level at higher abstractions makes it possible to construct the *Augmented-SFG* at higher levels of abstractions. In this example, the quantization noise sources b_2^1, b_2^2, b_1^3 and b_1^2 are augmented to the leaf-level sub-system graph at the lowest level in the hierarchy. At one level higher

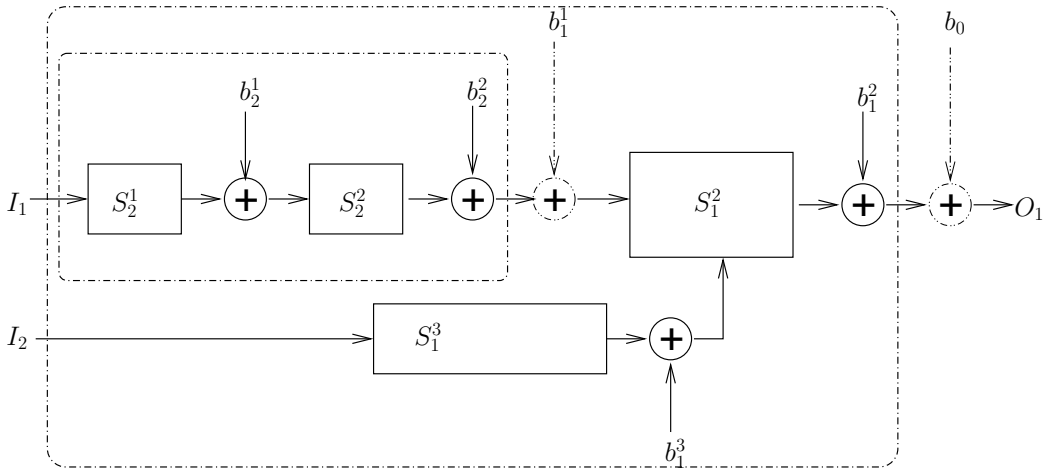


Figure 3.2: Augmented-SFG

in the hierarchy, noise sources b_2^1 and b_2^2 are no longer considered explicitly but are abstracted away as noise b_1^1. At the system-level, none of the sub-system noise sources are identified but for their combined effect as noise b_0.

3.1.2 Evaluating Output Quantization Noise

The focus of quantization noise analysis is on evaluation of the first order and second order moments of quantization noise at the system output in order to calculate the total output noise-power. Higher moments and other statistical properties are seldom an object of study. Given the properties of each quantization noise sources, the knowledge of the path function from each of the noise sources to the output is sufficient to calculate the total noise power. Therefore, in the classical analytical approach, the path function from the noise source to the output of the system H_b needs to be computed. In the hierarchical approach, this path function is equivalently obtained by considering the sub-system functions.

Consider calculating the impact of a noise source b in the sub-system S_2^1 on the system output from the example in Figure 3.1. The transfer function H_b corresponding to the path from noise source b to the output is equivalently given by transfer functions $H_{b_2^1}$, H_2^2 and H_1^2 in cascade as defined in the hierarchical system decomposition as shown in Figure 3.3.

Finding the transfer function from an operator source to the system output requires enumerating all the paths from the given operator to the output. In case of the classical approach, the procedure for enumerating all the paths has to be repeated as many times as the number of fixed-point operations. In the hierarchical approach, the noise power due to quantization from every fixed-point operation needs to be propagated only to the sub-system output individually. The path from the sub-system output to the system output is common and remains the same for all fixed-point operations within

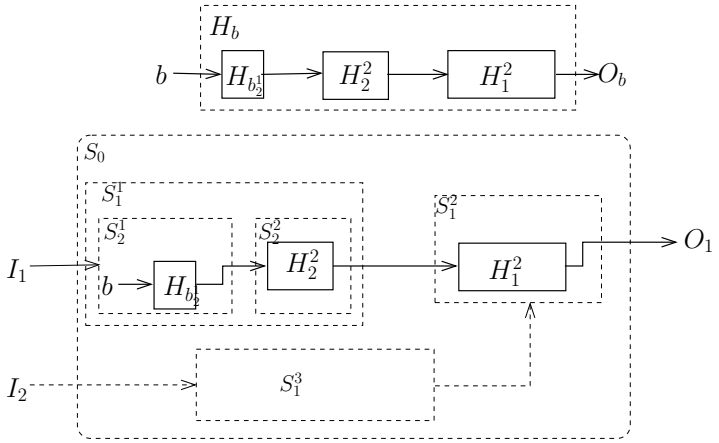


Figure 3.3: Propagation of noise b across hierarchies

the given sub-system. Therefore, the actual complexity of arriving at an expression for quantization noise power at the system-level output is greatly reduced.

In the example considered, if the transfer function $H_{b_1^2}$ is frequency selective, the spectral characteristics of the quantization noise due to noise source b at the output of the sub-system S_2^1 is already non-white. Moreover, the total noise at the output of sub-system S_2^1 comprises of contributions from all operator level noise sources in the system including contribution from noise source b. Therefore, if the path to the output across transfer functions H_2^2 and H_1^2 also happens to be frequency selective, mere calculation of the sub-system noise power is not sufficient. It is also necessary to know the power spectral density of quantization noise at the output of the sub-system S_2^1 for calculating the total output noise power.

Evaluate-Propagate-Add Strategy

An *evaluate-propagate-add* strategy which exploits the linearity property of noise propagation is used to calculate the output quantization noise of a given system. The noise contribution due to fixed-point operators in each of the sub-systems is *evaluated*. In practice, this amounts to the derivation of an analytical expression for calculating the total quantization noise-power at the output of each leaf-level sub-system participating in the SFG. This depends on the word-length assignments of the operators in the sub-system. The knowledge of other relevant parameters such as the spectral characteristics and the probability distribution function of the noise signal is essential to propagate the noise to the output of the system through the rest of the sub-systems. Derivation of all relevant parameters required for noise propagation is made as a part of the *evaluate* phase.

The quantization noise from each *single-noise-source* is injected into the system through the respective noise sources in the *Augmented-SFG* of the system and *propagated* to the output. That is, the noise generated in each of the sub-systems is consid-

ered one at a time and propagated through other sub-systems that lie on the path from the noise-source to the output to measure its impact on the total output quantization noise. After carrying out this process for every sub-system participating in the SFG of the system, the total quantization noise at the output is calculated by adding up the individual noise source contributions from every sub-system. Clearly, the use of this technique requires that the noise-propagation through each and every operator in the system should be modeled analytically. The propagation model of quantization noise is linear and it can be modeled with sufficient accuracy using a first order propagation model as given by Equation 2.14 in Section 2.3.2.

In systems whose signal flow graph has a feed-forward topology, a clear order of precedence already exists and this order dictates the order in which the noise generated within every sub-system has to be propagated. In case of systems with feed-back, the presence of loops introduces cyclic dependencies. In order to continue using the evaluate-propagate strategy, the augmented SFG cannot be used directly and it requires a transformation to make it feed-forward noise propagation SFG while respecting the sub-system boundaries.

Consider the graph shown in Figure 3.4. This graph shows four leaf-level sub-systems with sub-systems S_L^2 and S_L^3 connected with cyclic dependency. Let α and β be the transfer functions of the sub-system S_L^2 and S_L^3 respectively.

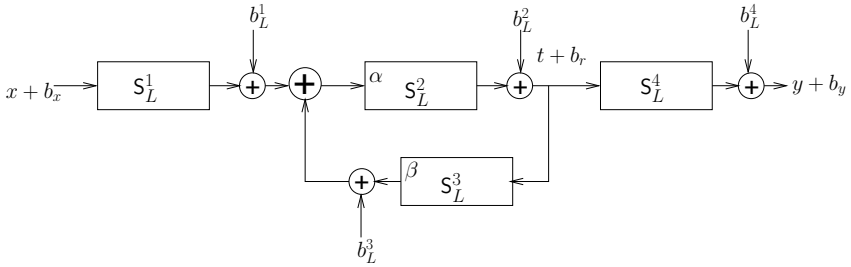


Figure 3.4: Augmented SFG of with cyclic dependency

Drawing from the standard result in feed-back control theory, the gain G of the system with forward gain α and feed-back gain β connected in a positive feed-back configuration is given as

$$G = \frac{\alpha}{1 - \alpha\beta}. \tag{3.1}$$

The passage of quantization noise through the sub-systems forming the loop is given by the path gain G defined in Equation 3.1. Exploiting the linearity property of quantization noise propagation models yet again, the noise introduced into the system is considered one by one. This effectively breaks up the cyclic dependencies and transforms the feed-back signal flow graph into a feed-forward noise-propagation SFG incorporating the quantization noise corresponding to each individual sub-system in the cyclic loop. Considering the injection points of these noise sources for the example in Figure 3.4, the equivalent noise propagation SFG is obtained as a feed-forward network as shown in Figure 3.5.

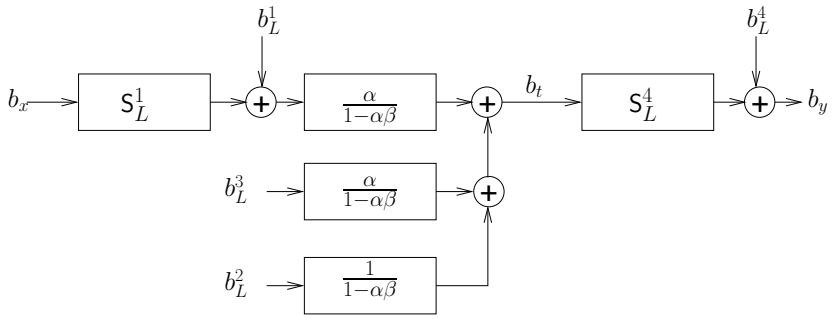


Figure 3.5: Noise propagation SFG for topology with cyclic dependency

3.2 The Single Noise Source Model

The total quantization noise measured at the output of any fixed-point sub-system in the signal flow graph of a given system has two sources. One of them is the noise generated by the fixed-point operators within the sub-system under consideration. This corresponds to the noise source added into the SFG to obtain the augmented SFG as described in the previous section. The second source is the noise generated due to fixed-point quantization noise generated in other sub-systems and propagated through the sub-system under consideration.

The pseudo-quantization noise (PQN) model proposed by Widrow for operator quantization noise statistically mimics the errors generated by fixed-point operators. In other words, the error injected into the system by a fixed-point operator can be generated by a random process with attributes defined by Widrow's model. By abstracting away all the operator level noise sources within a sub-system, the Single Noise Source (SNS) model extends the idea of PQN model by treating the total quantization noise generated within a given sub-system as a random process. Also, the SNS model provides a technique for transformation of the random process representing the quantization noise at the input of the sub-system under consideration to its contribution at the sub-system output. Therefore, the objective of the SNS model is to define the parameters of the random process such that it is statistically equivalent to the error due to quantization when fixed-point operations are used.

Consider, a sub-system B with input x and output y implemented using operators with infinite precision. It is expected that the input and output of this system is perturbed by the quantization noise when their fixed-point counter parts are used. When the conditions for PQN modeling for quantization noise are satisfied, the quantization noise and the signal are uncorrelated with one another. Therefore, as shown in Figure 3.6; the input quantization noise component b_x and the output quantization noise b_y are uncorrelated with the input signal x and output signal y respectively. The noise b_y is obtained as a sum of the noise resulting from the propagation of the noise b_x through the sub-system and propagation of noise generated within the sub-system b_g due to fixed-point operations. These two components are marked as b_s^t and b_s^g in Figure 3.7

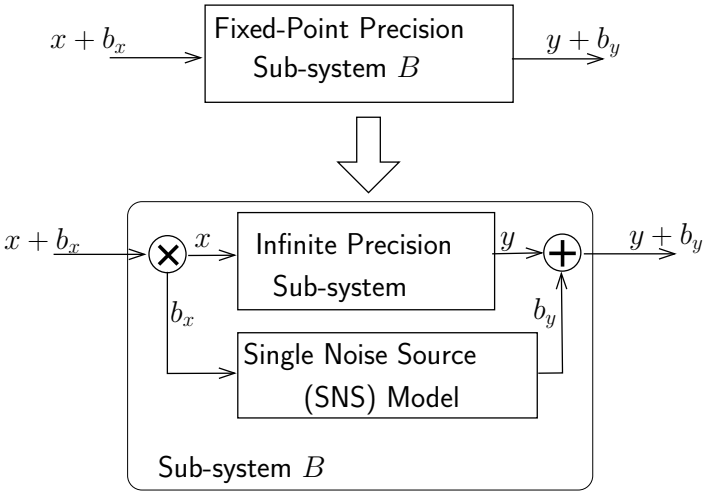


Figure 3.6: Abstraction of the Single Noise Source (SNS) model

3.2.1 Single Noise Source Model Parameters

Any random process is completely characterised by its power, spectral distribution and density distribution properties. Therefore, apart from knowing the noise-power at the output of a given sub-system, the SNS model also incorporates the noise power spectral density (PSD) and noise probability density function (PDF) properties. The knowledge of these properties is essential for propagation of quantization noise through sub-systems that are frequency selective (sub-systems with memory elements such as filters) and operators whose output is sensitive to the value of the signal (such as the zero detector or a QAM[1]-discriminator). In other words, the noise-power, PSD and PDF of the noise b_y should be such that they statistically match up with the fixed-point error properties obtained by fixed-point simulation.

The various parameters of the single noise-source model can be split into two categories depending on whether it is used for evaluation of the total quantization noise contribution of the block or it serves the purpose of propagating the input quantization noise. For the purpose of illustration, consider the SNS model for a single input, single output system is as shown in Figure 3.7. The schematic of the SNS parameters model has two branches: one corresponding to the noise contributed by the input quantization noise and the other branch corresponding to the noise generated within the given sub-system.

Noise Power: The noise-power b_g is representative of all the operator level noise-sources contained within the given sub-system. The value of the noise power of the source b_g is set to the sum of contribution of all individual noise sources within the fixed-point sub-system. The total noise variance is calculated by using the technique

[1]QAM: quadrature amplitude modulation

described in the Chapter 2. The noise power contribution due to the input noise b_x is similarly calculated by considering the functionality of the given sub-system and added separately to the total noise power b_y as

$$b_y = b_s^g + b_s^t. \tag{3.2}$$

Noise Power Spectral Density (PSD): From Widrow's PQN model, it is known that the operator level noise sources are spectrally white. However, the frequency selective path functions from each operator noise source modulates their spectral characteristics. Therefore, if the sub-system is frequency selective (i.e. it consists of delay elements), the noise generated from each of the operator sources is spectrally modulated. In other words, although the noise b_g is white; it is not necessary that the noise b_s^g continues to be white. The spectral properties of the noise b_S^g depends on the frequency selectivity of the paths in the sub-system and is captured by the *Generate Filter* block shown in Figure 3.7. This filter is derived out of the sub-system functionality and essentially defines a transfer function which can be analytically evaluated to calculate the power spectral density of the output quantization noise. Similarly, the spectral effects on the noise b_x associated with the input signal x is captured by the *Transmit Filter* in the SNS model to generate the signal b_s^t. This filter is the response of the sub-system to input quantization noise and it is practically defined by the magnitude spectrum of the system transfer function. Techniques for determination of the various auto and cross noise spectral distributions are discussed in Section 3.3.

Probability Density Function (PDF): The quantization noise at the operator source is known to be uniformly distributed according to Widrow's quantization noise model. At the output of any given sub-system, a number of such independent or uncorrelated with one another, scaled and uniformly distributed noise variables are added together. The shape of the output quantization noise depends on the relative noise-powers of each of the sources and their path functions. Under the assumption that distribution of errors is unimodal, the 4^{th} moment of the quantization noise is chosen as the metric or a measure of its PDF. The *Noise Shaping* block shown in

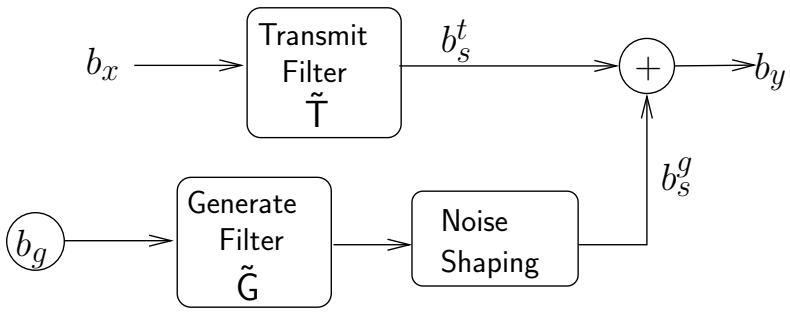


Figure 3.7: Inside the single noise source(SNS) model

Figure 3.7 captures this parameter at the output of any given sub-system. Analytical techniques for determining the 4^{th} moment of the noise generated within the sub-system at its output and techniques to propagate it through the rest of the sub-systems to the system output is discussed in Section 3.4.

Simulating the Random Process

It has to be noted here that the single noise source model captures the various parameters of the random process to accurately characterize the quantization noise. Deriving these parameters analytically is considered in the rest of this chapter. The schematic in Figure 3.7 can be used when either the quantization noise is spectrally white or the distribution can be approximated to be a Gaussian. In scenarios when neither of the two conditions hold, techniques such as [60] describe methods for actual generation of the random processes with specified spectral and distribution characteristics. Discussion of these techniques for the purpose of generation of noise is out of scope of this thesis.

Multiple Input Multiple Output Case

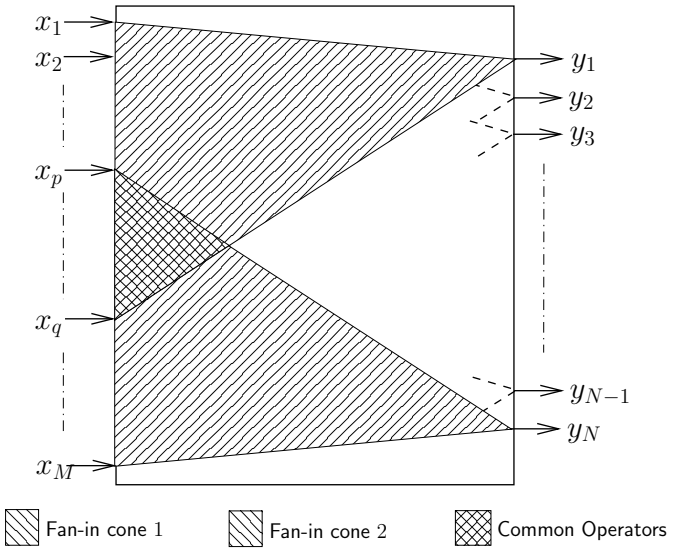


Figure 3.8: Subsystem with multiple inputs and outputs

Without loss of generality, we can consider the application of SNS model for a sub-system with M inputs and N outputs outputs as shown in Figure 3.8. In such a sub-system, the quantization noise at each of N outputs is influenced by one or more inputs. Moreover, the outputs share various operators and data-paths between them. The first output y_1 is fed from some or all of the inputs starting from x_1 through x_q. The last output y_N are fed from some or all of the inputs starting x_p through x_M.

Each of the outputs is affected by all the operators that lie in the path from each of the inputs to the respective output. These operators are considered a set and are referred to as Fan-in cone or simply *cone*[1].

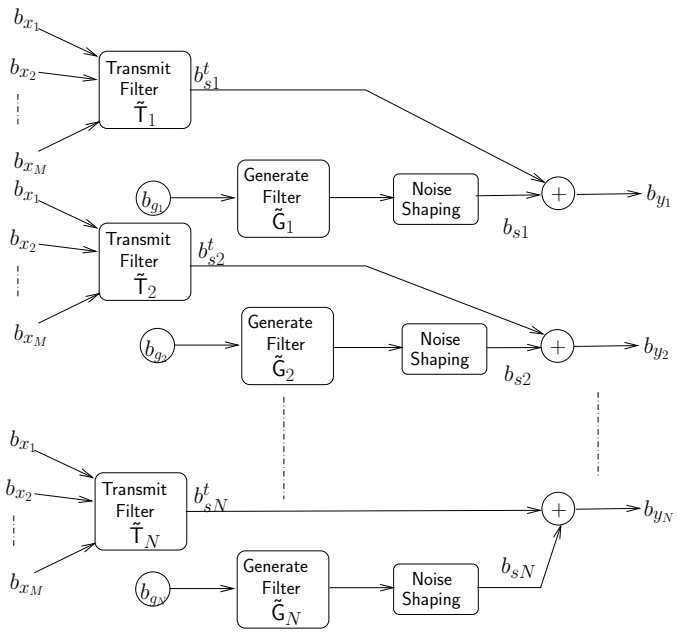


Figure 3.9: Single Noise Source Model for Multi Input Multi Output sub-system

The components of the SNS model for such a multi-input, multi-output sub-system is shown in the Figure 3.9. The SNS model essentially treats such systems as a collection of many multiple-input, single-output systems. The quantization noise at any of the outputs is obtained by adding up the contribution of each of the operator in the respective cones. The power, spectral characteristics and the shape of the total noise is captured by the respective b_g, *Generate filter* and Noise Shaping blocks SNS components. The difference lies in the *Transmit Filter* part. As several inputs contribute to the output, the transmit filter is a multi-input filter. The inputs coming into the *Transmit Filter* could be correlated with one another. So, while deriving the transfer function, special attention for this has to be given. This is repeated for every set of operator cones to obtain the total quantization noise at the respective outputs to obtain the SNS model for the sub-system with multiple-input and multiple-output. Clearly, there are as many SNS component sets as the number of outputs.

Cross Correlation In Figure 3.8, the area shared between two output cones (shaded along both directions) represents those operators participating in computation of outputs y_1 and y_N. If some of those operators are a source of quantization noise, then

[1]This is a popular terminology in the area of logic circuit analysis and is used here in a similar sense. The difference is just that the operators take the place of logic gates.

the total quantization noise at the output are correlated with one another. Therefore, the cross correlation between any pair of outputs needs to be calculated to completely define the output quantization noise of the given multi-output sub-system. Hence, the calculation of *Cross Spectrum*[1] is also very much a part of the SNS model.

3.3 Estimating Spectral Properties

In this section, the spectral components of the SNS model that affect the spectral characteristics of the output quantization noise at each of the outputs is studied in more detail. To begin with, a technique for deriving both *Transmit* and *Generate* filters in case of a given computational sub-system is studied. Analytical expressions for deriving the autocorrelation spectra of every output is studied in case of both LTI and non-LTI systems. This is extended to cover the derivation of cross-correlation spectra of any pair of signals at the output of the sub-system under consideration. The computational complexity of deriving these filters is estimated. Using the linear noise propagation model with respect to the input quantization noise as described in Chapter 2, it is possible to calculate the noise at the i^{th} output as

$$b_{y_i}(n) = \underbrace{\sum_{q=1}^{M} h_q^i(n) \star b_{x_q}(n)}_{Transmit\ Filter} + \underbrace{\sum_{p=0}^{P} h_p^i(n) \star b_{g_p}(n)}_{Generate\ Filter}, \tag{3.3}$$

where $h_q^i(n)$ and $h_p^i(n)$ are the impulse responses from the q^{th} input to the i^{th} output and from the p^{th} noise source emanating from one of the operators within the sub-system to the i^{th} output respectively. Also, $\star$ represents the linear convolution operation. $b_{x_q}(n)$ is the noise associated with the q^{th} input and $b_{g_p}(n)$ is the noise generated from the p^{th} operator noise source.

The *Transmit Filter* and *Generate Filter* terms contributing to the output quantization noise are marked in Equation 3.3. Due to linear noise propagation models, the nature of both *Transmit Filter* and *Generate Filter* is similar. The only difference that need to be considered is that the inputs to the *Transmit Filter* can be correlated with one another whereas the operator noise sources are independent and their effect is captured as one single noise source at the input of the *Generate Filter*.

3.3.1 Estimating Auto-Correlation Spectrum (PSD)

Consider estimating $b_y(n)$, the noise signal at the i^{th} output of a sub-system with M inputs. Taking into account the similarities between the *Transmit Filter* and *Generate Filter* blocks, the source of quantization noise can either be the noise presented at the input or the noise generated due to a fixed-point operator. Let $h_q(n)$[2] be the time-varying impulse response of the path function from the q^{th} noise source to the

[1] Frequency domain representation of Cross Correlation between two signals

[2] The index i is dropped for the sake of simplicity of notation.

i^{th} output. To account for noise propagation through time varying and non-linear systems, the pseudo impulse response h_q can be time varying in nature (as discussed in Chapter 2). The total quantization noise defined in Equation 3.3 after expanding the convolution is the sum effect of all quantization noise sources at the output. It is then written as

$$b_y(n) = \sum_{q=1}^{M} \sum_{k=-\infty}^{\infty} h_q(k,n) b_q(n-k), \tag{3.4}$$

where $h_q(k,n)$ is the value of the k^{th} coefficient at any time n of the time-varying impulse response $h_q(n)$. In case of an LTI system, the value of this coefficient does not vary and is simply written as $h(k)$. The noise power spectral distribution at the sub-system output is the magnitude response obtained by calculating the Fourier transform of the autocorrelation function. The auto-correlation function of the i^{th} output quantization noise is given as

$$\begin{aligned}
R(m) &= E[b_y(n) b_y^*(n+m)] \\
&= E\left[\left\{\sum_{q=1}^{M}\sum_{k=-\infty}^{\infty} h_q(k,n) b_q(n-k)\right\}\left\{\sum_{r=1}^{M}\sum_{l=-\infty}^{\infty} h_r^*(l,n+m) b_r^*(n+m-l)\right\}\right] \\
&= \sum_{q=1}^{M}\sum_{r=1}^{M} E\left[\left\{\sum_{k=-\infty}^{\infty} h_q(k,n) b_q(n-k)\right\}\left\{\sum_{l=-\infty}^{\infty} h_r^*(l,n+m) b_r^*(n+m-l)\right\}\right] \\
&= \sum_{q=1}^{M}\sum_{r=1}^{M} R^{qr}(m),
\end{aligned} \tag{3.5}$$

where $R^{qr}(m)$ is the correlation between the contributions to the i^{th} output from the q^{th} and the r^{th} noise sources as shown in Figure 3.10. It has to be noted here that this requires the evaluation of autocorrelation and cross-correlation terms between the various time varying coefficients of the impulse response along the q^{th} and the r^{th} path.

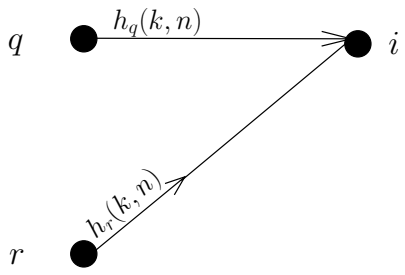


Figure 3.10: Estimating cross-correlation spectra between any two pairs

Considering all the coefficients of the time-varying impulse response $h_q(k,n)$ to be stationary, the i^{th} output noise power spectral density can be computed from the Fourier transform of $R(m)$ and is written as

$$\begin{aligned} S(e^{j\omega}) &= \sum_{m=-\infty}^{\infty} R(m)e^{-j\omega m} \\ &= \sum_{m=-\infty}^{\infty}\sum_{q=1}^{M}\sum_{r=1}^{M} R^{qr}(m)e^{-j\omega m} \\ &= \sum_{q=1}^{M}\sum_{r=1}^{M} \underbrace{\sum_{m=-\infty}^{\infty} R^{qr}(m)e^{-j\omega m}}_{S^{qr}(e^{j\omega})}. \end{aligned} \tag{3.6}$$

The output quantization noise PSD can thus be calculated by superposition of the PSD calculated from any two quantization noise inputs from q^{th} and r^{th} noise sources.

Consider evaluating the quantization noise PSD $S^{qr}(e^{j\omega})$. In terms of its correlation function, it can be expanded as

$$\begin{aligned} S^{qr}(e^{j\omega}) &= \sum_{m=-\infty}^{\infty} R^{qr}(m)e^{-j\omega m} \\ &= \sum_{m=-\infty}^{\infty} E\left[\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} h_q(k,n)h_r^*(l,n+m)b_q(n-k)b_r^*(n+m-l)\right]e^{-j\omega m}. \end{aligned} \tag{3.7}$$

From the PQN model, it follows that the quantization noise and the signal are uncorrelated. Thus, the partial output PSD $S^{qr}(e^{j\omega})$ can be written as

$$S^{qr}(e^{j\omega}) = \sum_{m=-\infty}^{\infty}\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} \underbrace{E[h_q(k,n)h_r^*(l,n+m)]}_{R^{qr}_{h_k h_l}(m)} \underbrace{E[b_q(n-k)b_r^*(n+m-l)]}_{R_{b_q b_r}(k+m-l)} e^{-j\omega m}, \tag{3.8}$$

where $R^{qr}_{h_k h_l}(m)$ is the correlation function between the values assumed by the k^{th} and the l^{th} coefficients of $h_q(n)$ and $h_r(n)$ respectively and $R_{b_q b_r}(m)$ is the cross correlation function between the q^{th} and r^{th} input quantization noise sources. The time varying coefficients of the impulse response $h_q(k,n)$ can be written as a sum of zero mean random process by separating the mean of the random process explicitly as

$$h_q(k,n) = \mu_q(k) + \tilde{h}(k,n), \tag{3.9}$$

where $\mu_q(k)$ and $\tilde{h}_q(k,n)$ corresponds to the mean of the random process of the k^{th} coefficient of the time varying impulse response $h(k,n)$ and the zero mean time varying components respectively. Assuming that the random process corresponding to coefficients of time-varying impulse response is stationary, the mean value of the coefficients does not change with time and are therefore a constant. Let $\mu_q(k)$ and $\mu_r(l)$ be the mean value of the k^{th} and the l^{th} coefficients in the time-varying impulse response for path q and path r respectively. The correlation between the k^{th} and l^{th} coefficients can then be written and expanded as

$$\begin{aligned} R^{qr}_{h_k h_l}(m) &= E[\{\bar{h}_q(k,n)+\tilde{h}_q(k,n)\}\{\bar{h}^*_r(l,n+m)+\tilde{h}^*_r(l,n+m)\}] \\ &= \mu_q(k)\mu^*_r(l) + E[\tilde{h}_q(k,n)\tilde{h}^*_r(l,n+m)] \\ &\quad +\mu_r(l)\underbrace{E[\tilde{h}_q(k,n)]}_{=0} + \underbrace{E[\tilde{h}^*_r(l,n+m)]}_{=0}. \end{aligned} \tag{3.10}$$

By substituting the expression for $R^{qr}_{h_k h_l}(m)$ in Equation 3.8, the autocorrelation function at the i^{th} output due to contribution from the noise source q and r can be written as

$$\begin{aligned} S^{qr}(e^{j\omega}) &= \underbrace{\sum_{m=-\infty}^{\infty}\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}\mu_q(k)\mu^*_r(l)R_{b_q b_r}(k+m-l)e^{-j\omega m}}_{\bar{S}_{qr}(e^{j\omega})} \\ &\quad + \underbrace{\sum_{m=-\infty}^{\infty}\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}E[\tilde{h}_q(k,n)\tilde{h}^*_r(l,n+m)]R_{b_q b_r}(k+m-l)e^{-j\omega m}}_{\tilde{S}_{qr}(e^{j\omega})}. \end{aligned} \tag{3.11}$$

The expression in Equation 3.11 for autocorrelation essentially consists of two terms. $\bar{S}_{qr}(e^{j\omega})$ corresponds to the average value of the coefficients and $\tilde{S}_{qr}(e^{j\omega})$ corresponds to the coefficients with zero mean random process.

By considering $p = k+m-l$, the expression for $\bar{S}_{qr}(e^{j\omega})$ can be written as

$$\begin{aligned} \bar{S}_{qr}(e^{j\omega}) &= \bar{H}_q(e^{j\omega})\bar{H}^*_r(e^{j\omega})\sum_{p=-\infty}^{\infty}R_{b_q b_r}(p)e^{j\omega p} \\ \bar{S}_{qr}(e^{j\omega}) &= \bar{H}_q(e^{j\omega})\bar{H}^*_r(e^{j\omega})S_{b_q b_r}(e^{j\omega}), \end{aligned} \tag{3.12}$$

where the transfer function $\bar{H}(e^{j\omega})$ and $\bar{H}^*(e^{j\omega})$ are obtained by performing the DTFT[1]

[1] Discrete Time Fourier Transform

on the average of the values of each of these coefficients as

$$\bar{H}_k(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \mu_k(n) e^{-j\omega n}.$$

The result obtained in Equation 3.12 corresponds to the standard result obtained in case of LTI systems.

The time varying coefficients in Equation 3.11 need not be independent. By choosing to use equivalent expressions in the frequency domain representation, the evaluation of the time varying part can be simplified as it also makes use of the symmetry properties of correlation coefficients. The indexes k and l correspond to the k^{th} and the l^{th} coefficients respectively.

Consider the contribution of the time varying coefficients to the output noise power density spectrum:

$$\tilde{S}_{qr}(e^{j\omega}) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} E[\tilde{h}_q(k,n)\tilde{h}_r(l,n+m)] R_{b_q b_r}(k+m-l) e^{-j\omega m}. \quad (3.13)$$

For every coefficient pair of coefficients (k,l), the coefficient pair (l,k) needs to be considered to evaluate $\tilde{S}_{qr}(e^{j\omega})$. Taking the difference between the coefficient indices as $\Delta = |l-k|$, the expression for the total output noise spectrum can be split into three cases and can be written as

$$\tilde{S}_{qr}(e^{j\omega}) = \tilde{S}^1_{qr}(e^{j\omega}) + \tilde{S}^2_{qr}(e^{j\omega}) + \tilde{S}^3_{qr}(e^{j\omega}). \quad (3.14)$$

The first case corresponds to the correlation of impulse response coefficients when the indices (l,k) are equal. The first component can then be written as

$$\tilde{S}^1_{qr}(e^{j\omega}) = \sum_{k=-\infty}^{\infty} \underbrace{\sum_{m=-\infty}^{\infty} R_{\tilde{h}^k_q \tilde{h}^k_r}(m) R_{b_q b_r}(m) e^{-j\omega m}}_{\tilde{H}^{kk}_{qr}(e^{j\omega}) \star S_{b_q b_r}(e^{j\omega})}. \quad (3.15)$$

The second case is when the indices (l,k) are chosen such that $(l-k)$ is positive. The second component can then be written as

$$\tilde{S}^2_{qr}(e^{j\omega}) = \sum_{k=-\infty}^{\infty} \sum_{\Delta=1}^{\infty} \underbrace{\sum_{m=-\infty}^{\infty} R_{\tilde{h}^k_q \tilde{h}^{k+\Delta}_r}(m) R_{b_q b_r}(m-\Delta) e^{-j\omega m}}_{\tilde{H}^{k,k+\Delta}_{qr}(e^{j\omega}) \star S_{b_q b_r}(e^{j\omega}) e^{-j\omega\Delta}}. \quad (3.16)$$

The third case is when the indices (l,k) are chosen such that $(l-k)$ is negative. The third component can then be written as

$$\tilde{S}^3_{qr}(e^{j\omega}) = \sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty}\underbrace{\sum_{m=-\infty}^{\infty} R_{\tilde{h}_q^{k+\Delta}\tilde{h}_r^{k}}(m)R_{b_qb_r}(m+\Delta)e^{-j\omega m}}_{\tilde{H}_{qr}^{k+\Delta,k}(e^{j\omega})\star S_{b_qb_r}(e^{j\omega})e^{j\omega\Delta}}. \tag{3.17}$$

In the above equations for $\tilde{S}^1_{qr}(e^{j\omega}), \tilde{S}^2_{qr}(e^{j\omega})$ and $\tilde{S}^3_{qr}(e^{j\omega})$, the terms $S_{b_qb_r}(e^{j\omega})$ is the cross correlation spectrum between the noise signals from sources q and r. $\tilde{H}^{k,l}_{qr}(e^{j\omega})$ is the cross correlation spectrum between the k^{th} and the l^{th} coefficients of the impulse response $\tilde{h}_q$ and $\tilde{h}_r$ respectively and is given as

$$\tilde{H}^{k,l}_{qr}(e^{j\omega}) = \sum_{m=-\infty}^{\infty} R_{\tilde{h}_q^k\tilde{h}_r^l}(m)e^{j\omega}. \tag{3.18}$$

A commonly occurring condition is when there is only one input signal in a given sub-system or when the path from a given input to the output is not shared by other inputs. In such cases, there is a single path from the input source q to output i that needs to be considered. The output autocorrelation spectrum in this case can be written as $\tilde{S}^{qq}(e^{j\omega})$. The contribution to output power spectral density by zero mean time varying coefficient part is written as

$$\begin{aligned}\tilde{S}^{qq}(e^{j\omega}) &= \sum_{k=-\infty}^{\infty}\tilde{H}^{kk}_{qq}(e^{j\omega})\star S_{b_qb_q}(e^{j\omega}) + \sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty}\tilde{H}^{k,k+\Delta}_{qq}(e^{j\omega})\star S_{b_qb_q}(e^{j\omega})e^{-j\omega\Delta} \\ &\qquad + \left\{\sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty}\tilde{H}^{k,k+\Delta}_{qq}(e^{j\omega})\star S_{b_qb_q}(e^{j\omega})e^{-j\omega\Delta}\right\}^* \\ &= \sum_{k=-\infty}^{\infty}\tilde{H}^{kk}_{qq}(e^{j\omega})\star S_{b_qb_q}(e^{j\omega}) \\ &\qquad + 2Re\left\{\sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty}\tilde{H}^{k,k+\Delta}_{qq}(e^{j\omega})\star S_{b_qb_q}(e^{j\omega})e^{-j\omega\Delta}\right\}. \end{aligned}\tag{3.19}$$

The total output power spectral density $S^{qq}(e^{j\omega})$ is given by

$$\begin{aligned}S^{qq}(e^{j\omega}) &= \bar{H}_q(e^{j\omega})\bar{H}^*_q(e^{j\omega})S_{b_qb_q}(e^{j\omega}) + \sum_{k=-\infty}^{\infty}\tilde{H}^{kk}_{qq}(e^{j\omega})\star S_{b_qb_q}(e^{j\omega}) + \\ &\qquad 2Re\left\{\sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty}\tilde{H}^{k,k+\Delta}_{qq}(e^{j\omega})\star S_{b_qb_q}(e^{j\omega})e^{-j\omega\Delta}\right\}. \end{aligned}\tag{3.20}$$

When there are many inputs, the total expression $S^{qr}(e^{j\omega})$ is given as

$$\begin{aligned} S^{qr}(e^{j\omega}) &= \bar{H}_q(e^{j\omega})\bar{H}_r^*(e^{j\omega})S_{b_qb_r}(e^{j\omega}) + \sum_{k=-\infty}^{\infty} \tilde{H}_{qr}^{kk}(e^{j\omega}) \star S_{b_qb_q}(e^{j\omega}) + \\ &+ \sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty} \tilde{H}_{qr}^{k,k+\Delta}(e^{j\omega}) \star S_{b_qb_r}(e^{j\omega})e^{-j\omega\Delta} \\ &+ \sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty} \tilde{H}_{qr}^{k+\Delta,k}(e^{j\omega}) \star S_{b_qb_r}(e^{j\omega})e^{j\omega\Delta}. \end{aligned} \tag{3.21}$$

The total output autocorrelation spectrum $S(e^{j\omega m})$ is obtained by plugging the expression for $S^{qr}(e^{j\omega})$ in Equation 3.21 (which is equivalent to the expression in Equation 3.11) into Equation 3.6. The fact that for every pair of inputs signals (q, r), there exists two terms corresponding to (q, r) and then (r, q), the output autocorrelation power spectral density $S(e^{j\omega})$ is written as

$$S(e^{j\omega}) = \sum_{q=1}^{M} S^{qq}(e^{j\omega}) + \sum_{q=1}^{M}\sum_{r=q+1}^{M} \left\{S^{qr}(e^{j\omega}) + S^{rq}(e^{j\omega})\right\}. \tag{3.22}$$

Consider evaluating the expression for $S^{qr}(e^{j\omega}) + S^{rq}(e^{j\omega})$ with $q \neq r$. It can be obtained by suitably substituting for q and r in Equation 3.21. The total expression can be split as contribution from the average terms and the zero mean time varying random process. Then, using the complex conjugate property of the power spectral density[1], the expression for autocorrelation spectrum can be written as

$$\begin{aligned} S^{qr}(e^{j\omega}) + S^{rq}(e^{j\omega}) &= 2.Re\left\{\bar{H}_q(e^{j\omega})\bar{H}_r^*(e^{j\omega})S_{b_qb_r}(e^{j\omega})\right\} \\ &+2.Re\left\{\sum_{k=-\infty}^{\infty} \tilde{H}_{qr}^{k,k}(e^{j\omega}) \star S_{b_qb_r}(e^{j\omega})\right\} \\ &+2.Re\left\{\sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty} \tilde{H}_{qr}^{k,k+\Delta}(e^{j\omega}) \star S_{b_qb_r}(e^{j\omega})e^{-j\omega\Delta}\right\} \\ &+2.Re\left\{\sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty} \tilde{H}_{qr}^{k+\Delta,k}(e^{j\omega}) \star S_{b_qb_r}(e^{j\omega})e^{j\omega\Delta}\right\}. \end{aligned} \tag{3.23}$$

The expression for $S(e^{j\omega})$ in Equation 3.22 provides the final expression for evaluating the auto-correlation spectrum of the signal at any output. The total expression can be expanded by substituting Equation 3.20 for $S^{qq}(e^{j\omega})$ and Equation 3.23 for the expression $S^{qr}(e^{j\omega})) + S^{rq}(e^{j\omega})$. In the case of LTI systems, the coefficients do not

[1]Complex conjugate property: $S_{xy}(e^{j\omega}) = S_{yx}^*(e^{j\omega})$

not change with time. Therefore, the $\tilde{h}_q(k,n)$ does not exist. The PSD of the total quantization noise at the output is then written as

$$S(e^{j\omega}) = \sum_{q=1}^{M} |H_q(e^{j\omega})|^2 S_{b_q b_q}(e^{j\omega}) + 2.Re \sum_{q=1}^{M} \sum_{r=q+1}^{M} \left\{ \bar{H}_q(e^{j\omega}) \bar{H}_r^*(e^{j\omega}) S_{b_q b_r}(e^{j\omega}) \right\}. \tag{3.24}$$

3.3.2 Estimating Cross-Correlation Spectrum

The various sub-system outputs that share the operators between them can be correlated with one another. The study of cross correlation spectrum between pairs of the outputs become important when some computation is performed on such pairs of outputs outside of the sub-system. On the other hand, any pair of inputs in case of a sub-system with multiple inputs could be correlated. Estimating the cross-correlation spectrum thus becomes necessary when two inputs to a sub-system share at least one common noise source outside the sub-system.

If there are M_i number of sources contributing to the i^{th} output and M_j number of noise sources contributing to the j^{th} output, the total cross correlation between any two outputs i and j is given as

$$\begin{aligned}
R(m) &= E[b_{y_i}(n) b_{y_j}^*(n+m)] \\
&= E\left[\left\{ \sum_{q=1}^{M_i} \sum_{k=-\infty}^{\infty} h_{q^i}(k,n) b_q(n-k) \right\} \left\{ \sum_{r=1}^{M_j} \sum_{l=-\infty}^{\infty} h^*_{r^j}(l, n+m) b_r^*(n+m-l) \right\} \right] \\
&= \sum_{q=1}^{M_i} \sum_{r=1}^{M_j} \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h_{q^i}(k,n) h^*_{r^j}(l, n+m) b_q(n-k) b_r^*(n+m-l) \\
&= \sum_{q=1}^{M_i} \sum_{r=1}^{M_j} R^{q^i r^j}(m),
\end{aligned} \tag{3.25}$$

where $R^{q^i r^j}(m)$ is the cross correlation between the components of the output i and j contributed by the q^{th} and the r^{th} sources. The noise source q is propagated to the i^{th} output through the impulse response h_{q^i}. All the impulse responses considered between two sources (q,r) contributing to outputs (i,j) (i.e. $h_{q^i}, h_{r^i}, h_{q^j}$ and h_{r^j}) in the above expression are as shown in Figure 3.11.

The cross-correlation spectral characteristics $C(e^{j\omega})$ between outputs i and j is obtained by computing the Fourier transform of the cross correlation function $R(m)$ in

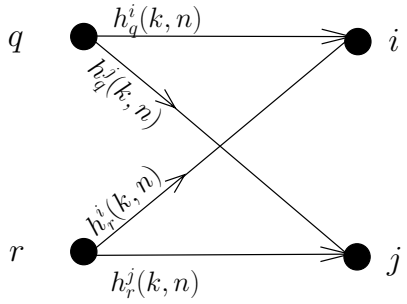


Figure 3.11: Estimating cross-correlation spectra between any two pairs

Equation 3.25. It is obtained as

$$\begin{aligned} C(e^{j\omega}) &= \sum_{m=-\infty}^{\infty} R(m)e^{-j\omega m} \\ &= \sum_{q=1}^{M_i}\sum_{r=1}^{M_j} \underbrace{\sum_{m=-\infty}^{\infty} R^{q^i r^j}(m)e^{-j\omega m}}_{C^{q^i r^j}(e^{j\omega})}. \end{aligned} \tag{3.26}$$

Following a procedure similar to that followed for calculating the auto correlation power spectral density, time-varying coefficients of the path functions are assumed to be stationary and they are treated as sum of mean value and a zero mean random process. The output cross correlation spectral density $C^{q^i r^j}(e^{j\omega})$ between any pair of outpus (i,j) due to inputs (q,r) is written as

$$\begin{aligned} C^{q^i r^j}(e^{j\omega}) &= \sum_{m=-\infty}^{\infty}\sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty} \underbrace{E[h_{q^i}(k,n)h^*_{r^j}(l,n+m)]}_{R^{q^i r^j}_{h_k h_l}(m)} \underbrace{E[b_q(n-k)b^*_r(n+m-l)]}_{R_{b_q b_r}(k+m-l)} e^{-j\omega m} \\ &= \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} E[\bar{h}_{q^i}(k,n)\bar{h}^*_{r^j}(l,n+m)]R_{b_q b_r}(k+m-l)e^{-j\omega} \\ &\quad + \sum_{k=-\infty}^{\infty}\sum_{l=-\infty}^{\infty}\sum_{m=-\infty}^{\infty} E[\tilde{h}_{q^i}(k,n)\tilde{h}^*_{r^j}(l,n+m)]R_{b_q b_r}(k+m-l)e^{-j\omega}, \end{aligned} \tag{3.27}$$

where the k^{th} and the l^{th} terms corresponds to the impulse response of the path from source q to output i and source r to j respectively. The contribution from time varying coefficients is split into three cases depending upon the indices (k,l) in order to be able to compute the spectrum in the Fourier domain as it was done for Equation 3.14. The

cross correlation spectrum is therefore given as

$$
\begin{aligned}
C^{q^i r^j}(e^{j\omega}) &= \bar{H}_{q^i}(e^{j\omega})\bar{H}^*_{r^j}(e^{j\omega})S_{b_q b_r}(e^{j\omega}) + \sum_{k=-\infty}^{\infty} \tilde{H}^{k,k}_{q^i r^j}(e^{j\omega}) \star S_{b_q b_r}(e^{j\omega}) \\
&+ \sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty} \tilde{H}^{k,k+\Delta}_{q^i r^j}(e^{j\omega}) \star S_{b_q b_r}(e^{j\omega})e^{-j\omega\Delta} \\
&+ \sum_{k=-\infty}^{\infty}\sum_{\Delta=1}^{\infty} \tilde{H}^{k+\Delta,k}_{q^i r^j}(e^{j\omega}) \star S_{b_q b_r}(e^{j\omega})e^{j\omega\Delta}, \qquad (3.28)
\end{aligned}
$$

where $\tilde{H}^{k,l}_{q^i r^j}(e^{j\omega})$ has the same definition as given in Equation 3.18.

The number of inputs M_i and M_j need not be the same and therefore, the cross spectral density need not be real. The total cross correlation spectral density $S_{ij}(e^{j\omega})$ is calculated by evaluating the Equation 3.28 for every combination of the pair (q, r) in Equation 3.26. The expression in Equation 3.26 cannot be simplified any further without making assumptions about the nature of input signals.

3.3.3 Cross-correlation Spectrum with Input Signals

Though the simple case of re-convergence can be handled easily by simply calculating the cross-correlation spectrum as discussed in the previous section, there can be scenarios where there is late re-convergence. Figure 3.12 shows one such scenario. In this scenario, the two outputs s^1_i and s^1_j emanating from the sub-system S_1 passes through sub-systems S_2, S_3 and S_4 respectively before re-converging into sub-system S_5.

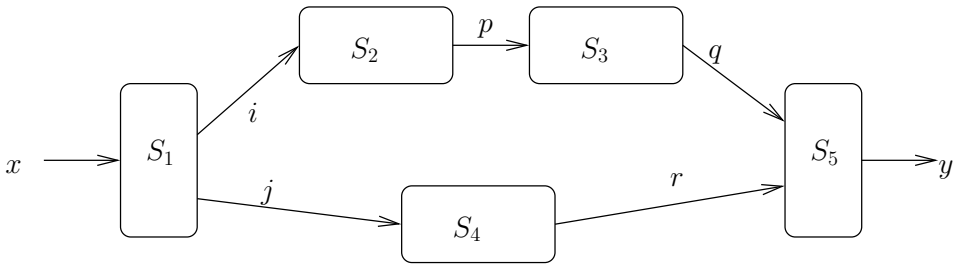


Figure 3.12: Illustrating re-convergence of signals in a hierarchical system

In a hierarchical method, it is required to characterize each subsystem thoroughly by means of the signal and quantization noise properties at its input and output. Such characterization should ideally be able to cover any usage scenario of the sub-system. In other words, it is important to be able to derive the required parameters from the sub-system parameters.

Consider calculating the cross correlation spectrum $S_{qr}(e^{j\omega})$ between the two inputs of the sub-system S_5 in Figure 3.12. To do this, it should be possible to calculate the cross correlation between the signals q and r by using the individual or mutual spectral

properties of any input signal or between any pair of input signals exposed by the sub-systems defined in the hierarchy. In order to do this, consider calculating the cross correlation between the output signal q and input signal p of the sub-system S_3. Let $h_3(n)$ be the time varying impulse response between the signals q and p. The cross correlation spectrum $S_{qp}(e^{j\omega})$ is given as

$$\begin{aligned} S_{qp}(e^{j\omega}) &= \sum_{m=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} E\left[h_3(k,n) b_p(n-k) b_p^*(n+m)\right] e^{-j\omega m} \\ &= \sum_{k=-\infty}^{\infty} \mu_{h_3}(k) e^{j\omega k} \sum_{m=-\infty}^{\infty} R_{b_p b_p}(k+m) e^{-j\omega(m+k)} \\ &= \bar{H_3}^*(e^{j\omega}) S_{b_p b_p}(e^{j\omega}), \end{aligned} \tag{3.29}$$

where $\bar{H}_3(e^{j\omega})$ is the Fourier transform of the time varying impulse response $h_3(n)$. Clearly, this cross correlation spectrum depends on the impulse response of the sub-system of the path function between signals p and q.

The input-output cross correlation spectrum for sub-systems S_4 and S_2 are similarly calculated and the cross correlation spectrum $S_{ij}(e^{j\omega})$ is calculated as described in the previous section. With this information, the cross correlation between the two inputs (q, r) is calculated by using the chain rule[1] for cross spectral densities as

$$S_{qr}(e^{j\omega}) = \frac{\left\{S_{qp}(e^{j\omega}) S_{pi}^*(e^{j\omega})\right\} S_{rj}^*(e^{j\omega})}{S_{pp}(e^{j\omega}) S_{ij}(e^{j\omega})}. \tag{3.30}$$

The simplicity of the above expression is that all the cross spectral density variables are obtained with local sub-systems of every sub-system. The above expression is valid only if the sub-systems transfer functions are mutually independent. If there is dependency, the cross correlation terms cannot be calculated this simply. Instead, it is more involved and takes the form of Equation 3.28 as derived in case of calculating cross spectral characteristics.

3.3.4 Complexity Analysis

The expressions for evaluating the correlation spectral densities of the signals as described in the previous sections are evaluated at discrete frequency points. Therefore, the number of FFT points used to evaluate these expressions determines the accuracy of the spectral characteristics. At the heart of expressions for determining spectral characteristics in Equations 3.20 and 3.26 is the computation of the FFT which is in turn used for calculating the convolution sum. The total time for calculating all the

[1] Chain rule: $S_{xy}(e^{j\omega}) = \frac{S_{xi}(e^{j\omega}) S_{yj}^*(e^{j\omega})}{S_{ij}(e^{j\omega})}$

spectral coefficients for a sub-system with M inputs and N outputs is given as

$$\begin{aligned} T &= \sum_{q=1}^{M}\sum_{r=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{q,r}^{i,j} \\ &= O(M^2.N^2.T_{q,r}^{i,j}), \end{aligned} \tag{3.31}$$

where $T_{q,r}^{i,j}$ corresponds to the time required to evaluate the correlation between the i^{th} and the j^{th} outputs due to inputs (q,r). When $i == j$, this corresponds to the time taken by the expression for autocorrelation. Otherwise, it is the time taken for calculating the cross-correlation terms.

Consider estimating one of the terms $T_{q,r}^{i,j}$. This essentially consists of two parts: i) the evaluation of the contribution by the mean of the coefficients $\bar{T}_{q,r}^{i,j}$ and ii) the contribution by the time varying component $\tilde{T}_{q,r}^{i,j}$. Therefore, $T^{i,i}$ can be written as

$$T_{q,r}^{i,j} = \bar{T}_{q,r}^{i,i} + \tilde{T}_{q,r}^{i,i}. \tag{3.32}$$

The time taken for the mean component is obtained as the total time taken for obtaining the expression for $\bar{H}_{q,r}(e^{j\omega})$ for each pair of inputs (q,r). To evaluate a given $\bar{H}_{q,r}(e^{j\omega})$, it is sufficient to calculate $H_q(e^{j\omega})$ and $H_r(e^{j\omega})$ that are obtained by calculating the N_{fft}-point FFT on the mean value sequence of the filter taps. This is then multiplied point-wise with the input correlation PSD. Therefore, the complexity of calculating $\bar{T}_{q,r}^{i,j}$ can be written as

$$\bar{T}_{q,r}^{i,j} = O(N_{fft}). \tag{3.33}$$

Similarly, the time taken by the time varying component is obtained as the total time required to derive $\tilde{H}_{q,r}^{k,l}$ for all coefficient pairs (k,l) occurring in the time-varying impulse response of each pair (q,r) of input paths. To obtain this, the cross correlation between the time varying filter coefficients $\tilde{h}_q^k(n)$ and $\tilde{h}_r^l(n)$ needs to be evaluated.

If N_s number of samples are used in a simulation-based approach, there are usually as many samples that need to be considered for each of the coefficients. Therefore, the computation of cross-correlation coefficient for each time-lag takes order of $O(N_s)$ in time. If this has to be repeated for all N_s time lags, the complexity of evaluating $R_{q,r}^{k,l}$ is of the order of $O(N_s^2)$. However, this sequence is used to calculate the N_{fft}-point FFT, where usually N_{fft} is much lesser than N_s. Therefore, not all N_s points are required to be calculated. In fact, a down sampled version of the correlation time lag sequence is sufficient to compute the N_{fft}-point FFT. Therefore, the effort for calculating the function $\tilde{H}_{q,r}^{k,l}(e^{j\omega})$ is given as $O(N_{fft}.N_s)$. However, the $O(N_s)$ arises out of computing the correlation between various time varying filter coefficients and it is a one time effort. Therefore, this is not the dominating factor for the over all time complexity of $\tilde{T}_{q,r}^{i,j}$. The function $\tilde{H}_{q,r}^{k,l}(e^{j\omega})$ is convolved with a scaled version of the input noise correlation PSD which. While scaling the input PSD takes $O(N_{fft})$ time, the circular convolution operation takes the time of order of $O(N_{fft}^2$. Suppose, if there are N_Q and N_R number

of coefficients on the q^{th} and the r^{th} path respectively, these operations are repeated $N_Q.N_R$ number of times. Therefore, the time $\tilde{T}^{i,j}_{q,r}$ is given as

$$\tilde{T}^{i,j}_{q,r} = O((N_{fft} + N^2_{fft}).N_Q.N_R). \tag{3.34}$$

By comparing Equations 3.33 and 3.34, it can be clearly seen that computation of the time varying coefficients have a higher time complexity. The contribution by the time-varying coefficients is large when the variance of the coefficients is comparable to the mean term. Owing to the high complexity of computing the contribution by the time-varying part, it may be ignored when the mean is sufficiently large.

3.3.5 Experiments with a Time Varying Filter

The expression for estimating auto-correlation power spectrum in Equations 3.21 and the expression for estimating the cross-correlation spectrum in Equation 3.28 have similar forms as they are obtained by calculating the cross correlation spectral density of the filter coefficients. A simple FIR filter with two-taps whose coefficients are time varying is chosen to evaluate these expressions. This FIR filter is representative of the path functions $h^q_i(e^{j\omega})$ between any noise source q and the i^{th} output participating in the evaluation of the cross correlation and auto correlation power spectrum.

Having more than one tap only increases the number of cross-correlation spectral terms to $\tilde{H}^{k,l}_{qr}(e^{j\omega})$. To keep the experiment simple and yet not losing generality, a two-tap filter as shown in Figure 3.15 with time varying coefficients is considered. The values taken by these coefficients are separated into the mean terms and the zero mean time varying terms. In order to be realisitic, a finite correlation between the zero mean time varying terms in introduced between the coefficients. The noise variance of these coefficients is comparable to their mean.

Since this example consists of one output only, it is enough if the Equation 3.20 is evaluated. It is required to calculate the cross-correlation spectra $H_{1,m}(e^{j\omega})$, $H_{m,1}(e^{j\omega})$ between the coefficients $h_1(n)$ and $h_m(n)$ and the auto-correlation spectra $H_{1,1}(e^{j\omega})$ and $H_{m,m}(e^{j\omega})$.

Each of the spectral evaluations of coefficient has two parts: i) corresponding to the mean and ii) corresponding to the time vayring part. The power spectral density of the quantization noise signal propagated through the two tap filter with 3 delay elements.

The Figure 3.14 shows the contribution by time varying coefficients with zero mean. It is clear that the analytical evaluation estimates the PSD smoothly whereas the PSD obtained by simulation broadly follows the trend set by the PSD curve that is obtained analytically.

To put this in perspective, consider the total PSD 3.15 at the output by taking into consideration the mean terms. The contribution to the total PSD due to the time varying part alone is labeled in the Figure 3.14 as *TV Coefficients Only*. The error between the PSD estimated by analytical technique and the actual PSD obtained by simulation is not large. In other words, estimation of quantization noise PSD using the

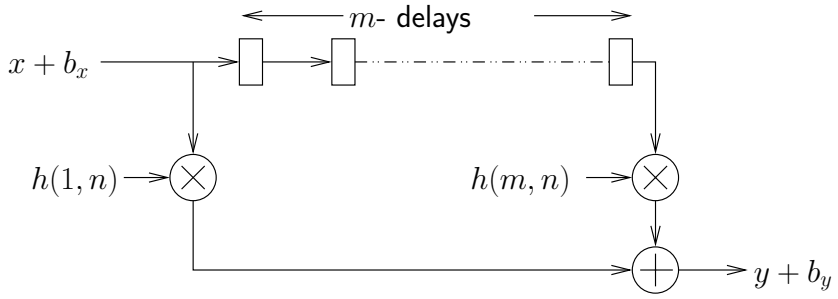


Figure 3.13: Schematic of a time varying FIR filter

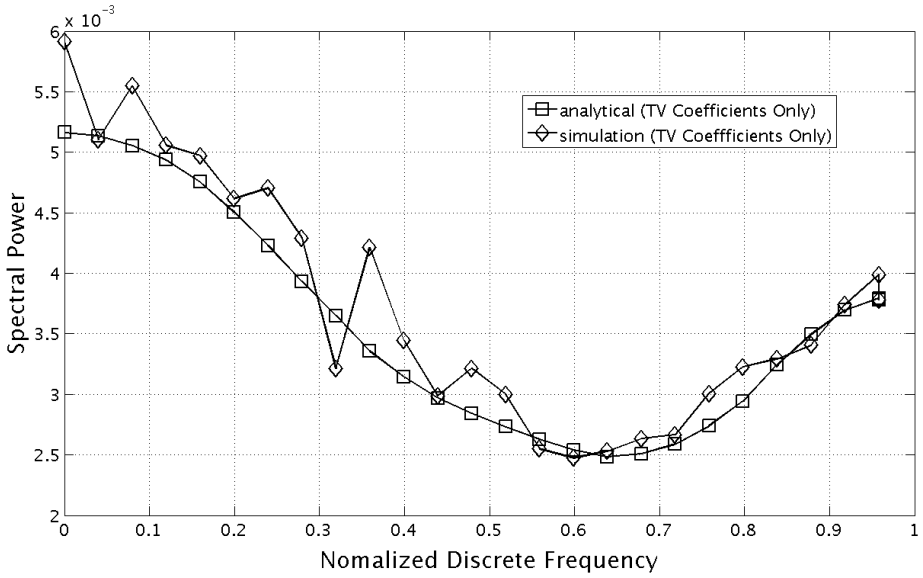


Figure 3.14: Output noise power spectral density, zero mean time varying coefficients

techniques presented in this section is fairly accurate.

3.4 Noise PDF Shaping

The probability density function (PDF) of quantization errors is the third parameter discussed in the context of the single noise source (SNS) model in Chapter 3.2.1. The PDF provides the information about the probability with which quantization error with a given magnitude would occur while performing fixed-point arithmetic. This information is useful in scenarios where it is required to calculate the probability of occurrence of an error with value greater than or lesser than a given threshold. Such scenarios commonly occur in many signal processing systems. The boundaries of QAM[1] decision

[1]Quadrature-Amplitude Modulation: a transmission scheme

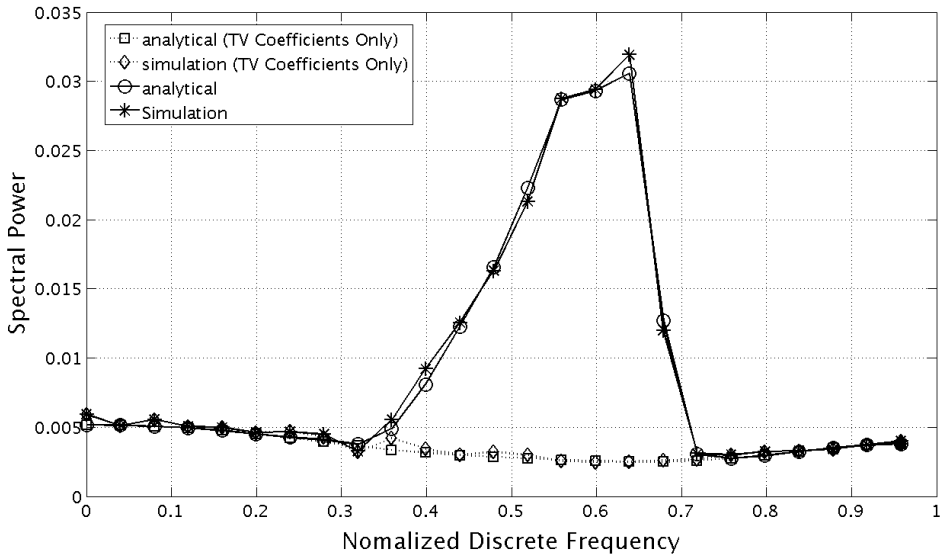


Figure 3.15: Output Noise Power Spectral Density, in perspective with non-zero mean time varying coefficients

operator, the saturation and over-flow boundaries of fixed-point adders, multipliers are common examples for such threshold values.

In this section, the probability density function (PDF) of the quantization noise at the output of a signal processing system is the subject of study. The noise shaping component is responsible for arriving at the exact PDF shape of the quantization noise at the output of sub-system. The shape of the PDF of any random signal is completely defined by its central moments. Analytical evaluation of the mean and variance of quantization noise is already known. In order to define the shape of the PDF, it is necessary to determine other higher moments. While it is impractical to evaluate all moments, it is well known that the shape of the PDF is predominantly defined by the first few moments.

Consider evaluating the third moment. Arguable, the third moment or the measure of skewness is mostly zero for quantization noise. The third moment is non-zero only when the distribution is asymmetric about its mean. There can be two scenarios where such asymmetry can occur. One of the scenarios is when there are multi-modal distributions or when the noise at source itself is asymmetric in nature. In statistics, multi-modal PDF occurs mainly by mixing population from various sets with different modes. In signal processing algorithm, which is predominantly data flow in nature (contrary to control and data flow), the output is usually an arithmetic function of various signals. Therefore, the quantization noise is a usually mapped to a unidimensional real or a complex number and not a mix of various quantization noise sources. Therefore, only those kind of systems whose quantization noise distribution is unimodal are considered in this thesis. The quantization noise source at the source is known to

be uniformly distributed. In Chapter 2, it was already discussed that the noise propagation model is linear. Therefore, it can be concluded that the noise at the output of a signal is always symmetric about its mean.

All higher odd moments of the quantization noise is zero when the quantization noise is symmetric and hence need not be considered in this discussion. The other higher moments would progressively add less information to the noise PDF as long as it is unimodal. Therefore, *Kurtosis* value of the distribution which is obtained as a function of the fourth moment is proposed to be used as a metric for quantifying the PDF shape.

3.4.1 A Motivating Example

A simple experiment highlights the importance of considering the quantization noise PDF. Consider a simple baseband transmitter receiver system as shown in Figure 3.16. A number of BPSK[1] symbols are transmitted through a static but noisy channel. At the receiver side, the signal is suitably amplified (so as to equalize the channel gain) and a BPSK discriminator is used to determine the received symbol. The discriminator output is compared with the transmitted signal to determine the symbol error rate.

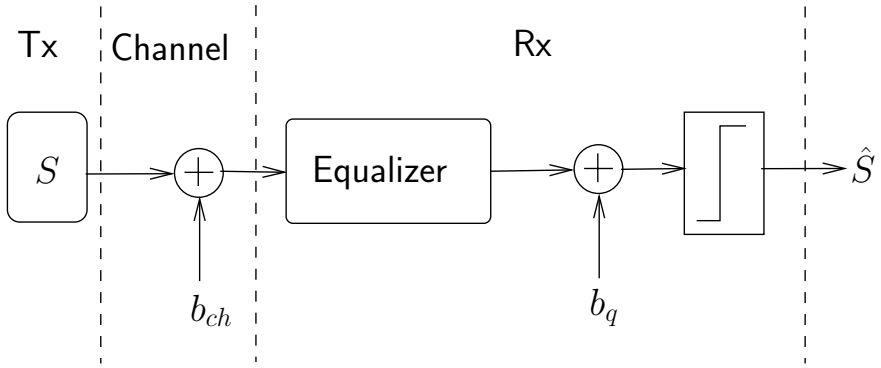


Figure 3.16: A baseband BPSK transmitter receiver experiment

In double precision, the BER is not zero due to the finite quantity of noise b_{ch} added by the channel. In case of a fixed-point receiver system, Figure 3.17 shows the BER plot for different quantization noise power b_q having the shape characterised by different Kurtosis values. The Gaussian shape has a Kurtosis value of 3 and the uniform distribution has a Kurtosis of 1.8. It is observed that the impact of considering the quantization noise shape is negligible when the actual quantization noise-power is very low and it tends to impact the result when the noise power is significantly large.

Therefore, the assumption that quantization noise is approximated by uniformly distributed by only considering the quantizer with the largest step-size or on the other hand considering it to be distributed with the shape of a Gaussian by attributing it to the central limit theorem may not always be accurate.

[1]Binary Phase Shift Keying: a transmission scheme

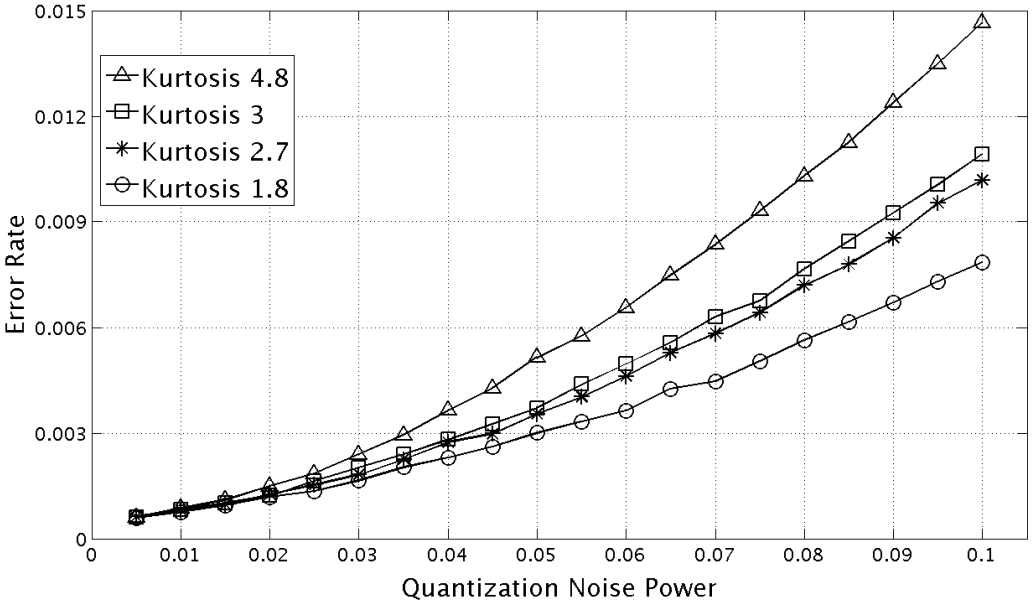


Figure 3.17: Impact of quantization noise PDF shape on BER

3.4.2 Evaluating Output Kurtosis

The noise propagation model in [87] is based on perturbation theory, which uses a linear model for noise propagation to compute the total output noise power. Every noise source (b_{g_j}), which is distributed uniformly, passes through a path function (H_j) to reach the output. The shape of the distribution is preserved if the path gain is a constant but changes according to the impulse response and the signal flow graph topology if the path is frequency selective.

Linear Combination of Input Noise Sources

Consider the evaluation of Kurtosis of the output quantization noise of a leaf-level sub-system. This requires the evaluation of the 4^{th} and the 2^{nd} moments. All noise sources due to fixed-point operation are spectrally white and uniformly distributed. If there are N operator level noise sources, consider evaluating $b_y(n)$: the total quantization noise generated within the sub-system. It is obtained as the sum of scaled and delayed versions of the quantization noise source due to each fixed-point operation as

$$\begin{aligned} b_y(n) &= \sum_{j=1}^{N} b_{y_j}(n) \\ &= \sum_{j=1}^{N} \sum_{k=-\infty}^{\infty} h_j(k,n) b_{g_j}(n-k), \end{aligned} \tag{3.35}$$

where $h_j(k,n)$ is the k^{th} time varying coefficient of the j^{th} path from noise source b_{g_j} to the output. By definition of the PQN model, the operator level noise sources are uncorrelated with one another. Therefore, the delayed version of the operator noise source are also remain uncorrelated.

The total quantization noise at the output is therefore a sum of scaled versions of the input quantization noise and can be written as

$$b_y(n) = \sum_{j=1}^{N} \sum_{k=1}^{M} b_{j,k}(n), \tag{3.36}$$

where $b_{j,k}$ corresponds to the j^{th} noise source scaled by the filter coefficient $h_j(k,n)$: the k^{th} filter coefficient which delays the signal by k-samples in time. If the system under consideration is LTI, the value of $h_j(k,n)$ is time invariant. Then, the uniform shape of the noise is preserved while the actual range of the noise is scaled depending upon the coefficient value. If the system under consideration is non-LTI, it is clear from the linear noise propagation model that the noise propagation path continues to be linear but is time varying. The uniform shape is more or less preserved if the variation of $h_j(l,n)$ is very small in comparison to the mean. Otherwise, the PDF value can even be *Leptokurtic*[1] in nature.

Analytical Evaluation

Consider evaluating the Kurtosis of the quantization noise at the output of a leaf level sub-system. The *Kurtosis* value of any distribution is defined as the ratio between μ_4, its fourth central moment and μ_2^2, square of its second central moment. Alternatively, *Excess Kurtosis* can also be used in place of Kurtosis and is defined to be numerically 3 lesser than the Kurtosis value. Excess Kurtosis is calculated as the ratio of κ_4, the fourth Cumulant and κ_2^2, the square of the second cumulant.

By using the additive property of cumulants [65], the fourth cumulant is easily obtained by scaling and adding the fourth cumulant of the noise at the source. The fourth cumulant and the second cumulant at the of the output quantization noise can then be written as

$$\begin{aligned} \kappa_4(b_y) &= \sum_{j=1}^{N} \sum_{k=-\infty}^{\infty} \kappa_4(b_{j,k}). \\ \kappa_2(b_y) &= \sum_{j=1}^{N} \sum_{k=-\infty}^{\infty} \kappa_2(b_{j,k}). \end{aligned}$$

The value $\kappa_4(b_{j,k})$ is obtained as a function of the moments of $b_j(n)$ and $h_j(l,n)$.

[1] Kurtosis of the distribution is greater than 3.

In case of LTI systems, the expression for cumulants can be simplified and written as

$$\kappa_4(b_y) = \sum_{j=1}^{N} \sum_{l=-\infty}^{\infty} h(l)^4 \kappa_4(b_j).$$
$$\kappa_2(b_y) = \sum_{j=1}^{N} \sum_{l=-\infty}^{\infty} h(l)^2 \kappa_2(b_j).$$

It has to be noted here that this simplification is valid only when the noise sources $b_j(n)$ are uncorrelated with one another and their spectral characteristics is white.

3.4.3 Experiments: Estimating PDF Shape

Quantization noise at the output of an FIR filter

In order to illustrate the use of analytical expression in Equation 3.36, a simple FIR filter is considered. This example corresponds to the propagation of quantization noise from one of the noise sources to the output of the sub-system. Consider an FIR filter with 32 taps. From Equations 3.37, the PDF shape of the output quantization noise of the filter is obtained propagating the Kurtosis and variance of a uniform random variable through the filter taps with 32 different powers. If all the coefficients were to take comparable values, the output noise shape was expected to be a Gaussian (obtained from central limit theorem). Let 4 of the coefficients be relatively large in comparison with the rest of the coefficients. The noise source $b_{j,k}$ corresponding to each of these coefficients have higher power and therefore alter the shape of the output quantization noise. In this example, the addition of PDFs with large noise power due to the coefficients with higher value makes the output quantization noise PDF *Mesokurtic* (i.e. Kurtosis less than 3). It has to be kept in mind that the quantization noise power is independent of its distribution characteristics.

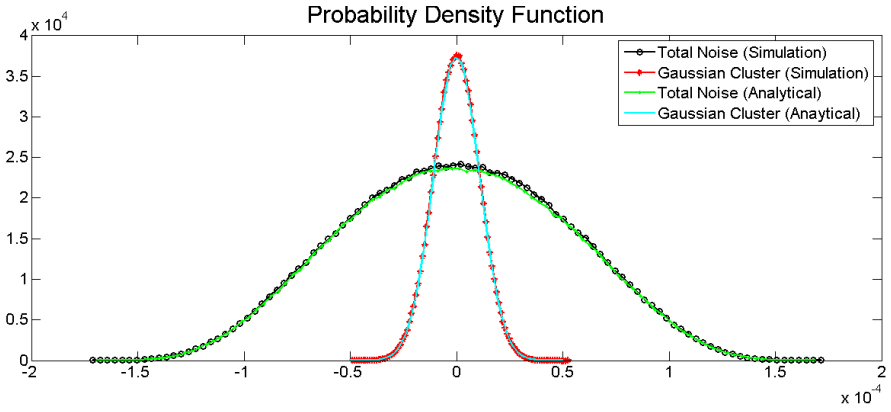


Figure 3.18: Computing PDFs obtained analytically and by simulation

The noise PDF at the output of the FIR filter is shown in Figure 3.18. The figure

also shows the Gaussian component in the signal. Since the output PDF is *Mesokurtic*, the output PDF can therefore be obtained as a weighted sum a Gaussian and a uniform random variable. If the output were to be *Leptokurtic* (i.e. Kurtosis greater than 3), the PDF can be obtained as a weighted sum of a suitable Leptokurtic distribution (such as a parametrized Laplacian) and the Gaussian. Another option is to obtain the desired Kurtosis of the PDF by using a parameterized generalized Gaussian distribution [98]. In Figure 3.18, the shape of the total noise obtained by simulation and that obtained analytically are matched very closely.

When the four coefficients with higher coefficient values are taken away from the summation, what remains is a sum of 28 identically distributed, uncorrelated random variables. Their sum is therefore a Gaussian. The Gaussian thus obtained by simulation and by analytical technique is shown also shown in the Figure 3.18. Comparing the total noise shapes with the Gaussian shape, if it were to be blindly assumed that the output is a Gaussian disregarding the actual shape, it is clear from the PDF plots that the error at the output of an un-smooth operator such as a discriminator placed at the mean value of the output PDF can have be sensitive to the input error PDF shape. This qualitatively explains the results of the experiment in the Section 3.4.1.

V-Blast Decoder

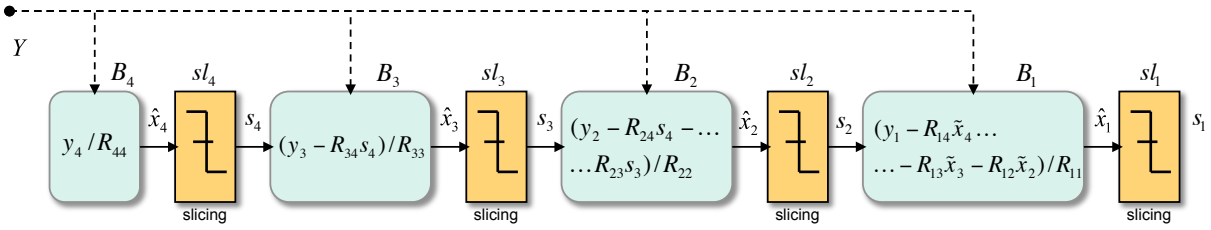


Figure 3.19: V-BLAST: data-flow model and associated smooth blocks

The V-BLAST algorithm [122] is the earliest of algorithms used for detection in the in MIMO[1] wireless systems. In this experiment, it is assumed that there are four transmit and four receive antennas. The signal flow graph in Figure 3.19 represents the flow of signals in a 4 receiver antenna V-BLAST detector. Each slicer decodes the output corresponding to one antenna. Since the symbols transmitted are decoded one after another, an error earlier on in this chain can affect the symbols detected later.

The path from the fourth antenna (block B_4) to the first (block B_1) consists of several arithmetic computations. The arithmetic blocks can be grouped together and the quantization noise can be modeled with the help of SNS model. The un-smooth decision operators at the output of every arithmetic cluster essentially determine the boundaries of the arithmetic clusters. None of the clusters have memory elements and therefore the output quantization noise in case of all clusters is white in noise density spectrum.

[1]MIMO: Wireless communication systems with multiple receiver and transmit antennas

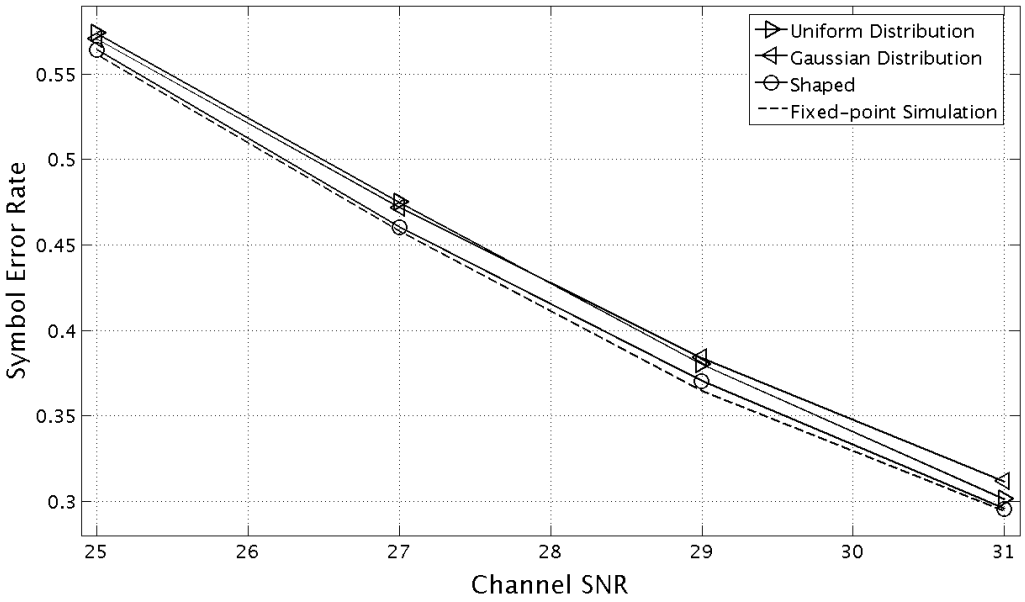


Figure 3.20: V-BLAST: Effect of quantization noise PDF shaping on BER

The quantization noise is generated with different Kurtosis values using the SNS models for each of sub-system separated by the un-smooth operator. Three PDF shapes are considered for conducting the experiment. The first two cases consider Gaussian and uniform PDF distribution for the quantization noise. The third case considers the PDF which has the shape confirming to the Kurtosis value. The total symbol error rate (SER) is shown in Figure 3.20. The SER obtained by using fixed-point simulation along with SER for various kurtosis is plotted for different channel noise conditions. The fractional bits in the fixed-point for all operations were assigned to $4-$bits. The kurtosis of the output quantization noise was obtained analytically. It has to be noted here that both R_{ij} and y_i are random variables and the Kurtosis of the output is hence usually more than 3. The Kurtosis at the output is obtained analytically for the non-linear division and multiplication operator by first obtaining the noise propagation expression for the output quantization noise and then taking the ratio of the moments defining Kurtosis. It can be observed that when the Single Noise source model is shaped with appropriate Kurtosis, the SER curve confirms more closely to the curve obtained by fixed-point simulation. Although the SER curves obtained by shaping the SNS model noise as uniform or Gaussian closely follows the trend obtained by fixed-point simulation, paying enough attention to the PDF shape brings the SNS model closer to the actual fixed-point simulation.

3.5 Application of the SNS model

Thus far, analytical formulas for deriving some of the statistical parameters of the quantization noise at the output of a given sub-system and its computational complexity were discussed. These parameters completely define the Single Noise Source (SNS) parameters at the sub-system-level. Owing to the high computation effort, deriving the entire SNS model often proves to be difficult. In this section, practical aspects of the application of SNS model for hierarchical accuracy evaluation are considered and an approach to selectively evaluate the SNS parameters based on the need of the sub-system characteristics is presented.

3.5.1 A Synthetic Example

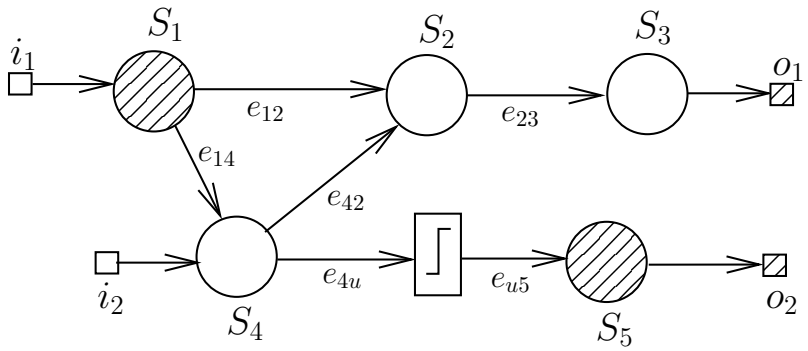


Figure 3.21: Application of the SNS model

Consider applying the SNS model on the signal flow graph (SFG) of the system whose sub-system topology is shown in the Figure 3.21. This example, represents a typical SFG one would encounter in a practical signal processing system although it is randomly constructed. There are five sub-systems and all the sub-systems are made of smooth operators. Sub-systems S_1 and S_5 are frequency selective (the corresponding nodes in the graph are shaded to indicate frequency selectivity) whereas sub-systems S_2, S_3 and S_4 do not change the frequency distribution of the noise power. Consequently, both outputs o_1 and o_2 display a frequency selective behavior. In other words, the quantization noise at both outputs need not be spectrally white. Therefore, in order to be consistent with the notation, they are also shaded. An un-smooth operator appears between the sub-systems S_4 and S_5.

To incorporate the fact that the input can also be quantized, the input pads are also considered to be nodes of the sub-system. The only difference between the inputs and the rest of the sub-systems is that while the SNS parameters of the leaf-level sub-systems are derived as a function of its functionality, the noise characteristics are either user defined or derived from data that are typically used for fixed-point simulation. As there could be more than one input to a given leaf-level sub-system, each input corresponds to a node in the SFG. Likewise, there can be more than one output for a given sub-

system and their spectral characteristics could be completely different. Therefore, each output pad is also considered as a node. In the example considered, there are two input nodes i_1, i_2 and two output nodes o_1, o_2. When it is not mentioned otherwise, the input quantization noise is typically considered to be white and uniformly distributed with the power determined by bits assigned for the fractional part.

3.5.2 Selective Evaluation of Spectral Parameters

Using the *evaluate-propagate-add* strategy discussed in Section 3.1.2, the quantization noise generated from within each of the sub-systems needs to be propagated through other sub-systems until the system output is reached. The determination of various noise properties of such single-noise-sources is driven by the sensitivities of the sub-systems through which a given noise source is passed.

In the above example, consider the quantization noise at the outputs of sub-system S_1. There are two outputs along the edges e_{12} and e_{14}. The noise signal corresponding to the output along the edge e_{12} is propagated through S_2 and S_3. Since both S_2 and S_3 are neither frequency sensitive nor sensitive to its level, it is sufficient if the quantization noise power at the output along the edge e_{12} is determined. Whereas, the output along the edge e_{14} passes through S_4, an un-smooth operator and the sub-system S_5. Clearly, the propagation of quantization noise is affected by both its frequency and PDF characteristics. Therefore, the determination of autocorrelation PSD and the PDF of the signal at the output of S_1 along the edge e_{14} is necessary. Also, the noise at the output of S_1 along the edge e_{14} re-converges onto S_2 after passing through S_4. Therefore, it is also necessary to calculate the cross correlation and input-output correlation between the signals corresponding to the noise signals along edges e_{12}, e_{14} and e_{14}, e_{42} respectively.

Similarly, consider the outputs of sub-system S_4 along edges e_{42}, e_{4u}. They do not converge anywhere in the sub-system graph and hence the cross correlation between the signals along these two edges is not necessary. The sub-system S_4 is not frequency sensitive and therefore, it is not necessary to calculate the spectral properties of the noise generated within or for the transmission of the signal. The input spectral characteristics is propagated without any modification. However, it is necessary to calculate the output PDF of the output along the edge e_{4u} as it feeds into the un-smooth operator.

The propagation of quantization noise through the un-smooth operator can be carried out by using simulation and the signal and noise characteristics at the output of the un-smooth operator can be determined by using the results obtained thereof. It is necessary to calculate the spectral characteristics at the output of the un-smooth operator as the sub-system S_5 is frequency selective.

Application of the SNS model to the above example is an illustration of the fact that the determination of all the SNS is not always necessary. Broadly, it can be said that the spectral characteristics of the noise single-noise-source need to be determined only if at least one of the sub-systems through which it needs to be propagated is frequency sensitive. Similarly, the distribution properties (PDF) of the noise signal needs to be studied only if un-smooth operators are encountered along the noise propagation path

and not otherwise. Therefore, unless mentioned otherwise, it can be assumed that the quantization noise is white and has a Gaussian distribution with a specified noise power that can be calculated by the techniques presented in Table 2.1. A data structure of the SNS parameters corresponding to every node is maintained and initialized to defaults. The data structure is as shown in Figure 3.22. In this figure, a sub-structure which refers to the selective evaluation of the spectral parameters of the node is highlighted. The structure essentially consists of a number of lists for the purpose of book keeping of the selective evaluations pertaining to its corresponding node. These flags are empty by default indicating that none of the spectral parameters need to be evaluated. The parameters of the SNS model corresponding to every sub-system is marked for evaluation and added onto the lists only if there is any expected deviation from this default behavior. In the following sections, procedures for selectively evaluating the various SNS parameters for a given SFG are defined. The spectral parameters and the distribution (noise shape) parameters are considered separately for evaluation.

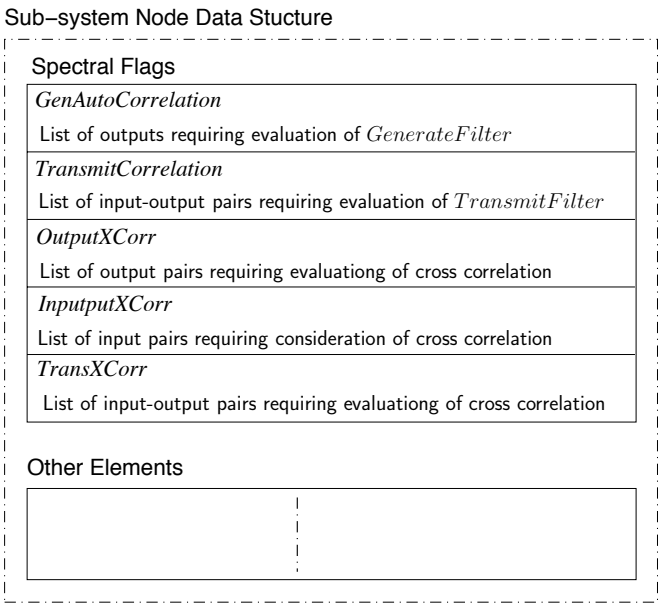


Figure 3.22: Sub-system Node data structure, selective spectral evaluation flags

A set of algorithms to mark these sepctral flags is detailed towards the end of this thesis in Appendix 6. In these algorithms are based on popular graph traversal algorithms. The need for evaluating various spectral parameters is determined by looking into the topology of signal flow graphs without needing any simulation.

3.5.3 Selective Determination of PDF Parameter

The PDF of quantization noise is not a matter of concern if the system does not contain any un-smooth operator. However, the presence of an un-smooth operator requires the total noise PDF at the input of all un-smooth operators be known. So,

clearly, only those sub-systems or more specifically those outputs of the sub-system whose paths terminate into the input of an un-smooth operator needs to be considered for evaluation of the noise PDF. The output of the un-smooth operator may be obtained by simulation or other alternative methods. It is unlikely that they can be evaluated analytically. Moreover, due to the non-linearity of the un-smooth operator, it is not possible to use the *evaluate-propagate-add* technique beyond one un-smooth operator. Therefore simulation is resorted to under such circumstances. More about this and the un-smooth effects are discussed in the next Chapter.

In the graph in Figure 3.21, the outputs of sub-systems S_1 and S_4 along edges e_{14} and e_{4u} only needs to be considered. Once the quantization noise shape has been determined at the input of the un-smooth operator, the un-smooth operator and the fixed-point accuracy of sub-system S_5 is obtained by simulation.

Calculating PDF at the Input of an Un-smooth Operator

The analytical expression for evaluation of Kurtosis at the output of the sub-system as discussed in Section 3.4 is applicable only when the noise is uncorrelated with itself in time or with other noise signals. This situation is unlike the spectral characteristics, where such correlations can be taken care of in the frequency domain. It is only in the case where the noise sources are emanating from various sub-systems are neither autocorrelated nor cross-correlated the 4^{th} cumulant of the total quantization noise due to all the contributing sub-systems can be added to obtain the total Kurtosis value analytically.

More generally, the quantization noise sources at the input of every *Generate Filter* is spectrally white. During the quantization noise evaluation, this noise is propagated through the *Generate Filter* followed by the *Transmit Filters* or path gains of other sub-systems that are encountered on the path to the system output. Taking advantage of the hierarchical decomposition, the individual path functions from every sub-system can be quickly determined as large parts of the path function are obtained from the *Transmit Filter* of the sub-systems in the path.

In the graph shown in Figure 3.21, let $G_1^4(e^{j\omega})$ be the *Generate Filter* of the sub-system corresponding to the node S_1 on its output along edge e_{14} and $T_4^{14}(e^{j\omega})$ be the path gain of the sub-system corresponding to node S_4 between its input along edge e_{14} and its output e_{4u}. Likewise, let $G_4^u(e^{j\omega})$ be the gain of the noise due to quantization in the sub-system corresponding to the node S_4. The spectral distribution of $G_4^u(e^{j\omega})$ is flat as there are no memory elements in the sub-system S_4. The total noise due to quantization $b_u(n)$ at the input of the smooth operator is given as

$$\begin{aligned} b_u(n) &= g_{14}(e^{j\omega}) \cdot G_1^4(e^{j\omega}) \cdot T_4^{14}(e^{j\omega}) + g_{4u}(e^{j\omega}) \cdot G_4^u(e^{j\omega}) \\ &= g_{14}(e^{j\omega}) \cdot H_1(e^{j\omega}) + g_{4u}(e^{j\omega}) \cdot G_4^i(e^{j\omega}), \end{aligned} \tag{3.37}$$

where $g_{14}(e^{j\omega})$ and $g_{4u}(e^{j\omega})$ are spectrally white noise sources corresponding to the total quantization noise at the operator sources in sub-systems S_1 and S_4 along the

path $S_1 \rightarrow S_4 \rightarrow$ (UnsmoothOperator). The transfer function $H_1(e^{j\omega})$ is obtained by multiplying the *Generate Filter* $G_1^4(e^{j\omega})$ and the *Transmit Filter* $T_4^{14}(e^{j\omega})$. With the availability of the time-varying impulse response whose input is a spectrally white noise source, the quantization noise PDF shape can be determined by the analytical technique to propagate 4^{th} cumulants presented in Section 3.4.

3.5.4 Examples: Application of Selective Evaluation

Using the information derived from the sub-systems interconnection topology, the kind of operators used to build the system and the knowledge of the input quantization noise properties, it is possible to carefully choose the parameters to be evaluated. Some examples are considered to illustrate the application of selective derivation of the SNS model parameters.

Fast Fourier Transform (FFT)

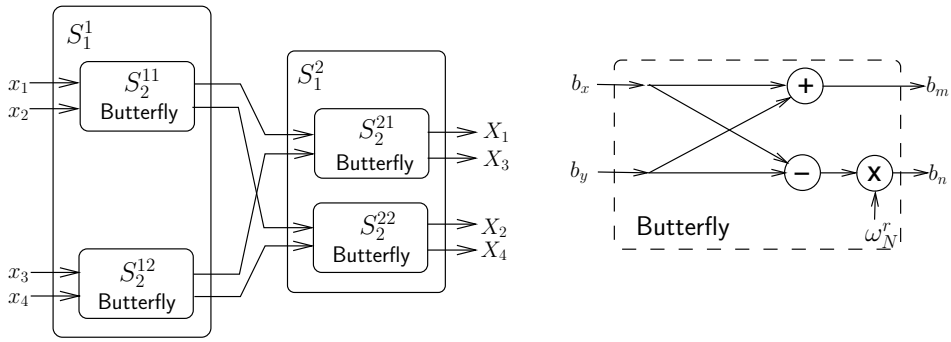


Figure 3.23: Hierarchical decomposition of a 4-point radix-2 FFT

Consider the 4-point radix-2 structure of the FFT algorithm as shown in Figure 3.23. It is customary to define the entire structure using the two-input two-output butterfly structure. Also, efforts to optimize the implementation of this popular algorithm such as [53] focuses on improving the butterfly structures. Therefore, it is easy to see the various butterfly structures as sub-systems in hierarchy to compute FFT as shown in Figure 3.23.

Consider the inputs and outputs of one of the sub-systems. The expression for output quantization noise is derived as

$$\begin{aligned} b_m &= b_x + b_y \\ b_n &= (b_x - b_y)\omega_N^r. \end{aligned} \tag{3.38}$$

Here, ω_N^r is referred to as the twiddle factor and is a constant. Following the discussion about constant multiplications in Section 2.3.3 and to keep the analysis simple, the twiddle factors are quantized in both infinite precision simulation and fixed-point simulation. The analytical expressions are compared against the errors obtained

by performing fixed-point simulation. The quantization noise power expressions are derived as

$$\mathrm{E}[|b_m|^2] = \mathrm{E}[|b_x|^2] + \mathrm{E}[|b_y|^2], \tag{3.39}$$
$$\mathrm{E}[|b_n|^2] = (\mathrm{E}[|b_x|^2] + \mathrm{E}[|b_y|^2]).|\omega'^r_N|^2, \tag{3.40}$$

where E[x] operator represents the expectation of the random variable x. The two outputs depend on both the inputs and are hence correlated with one another. However, both the outputs from of any sub-systems are never used together for any computation in the succeeding sub-systems. Therefore, it is not necessary to calculate the co-variance between the two outputs of the sub-system. Also, there are no memory elements or un-smooth operators. This means, that it is not necessary to derive any of the spectral and PDF parameters of the sub-system.

Band-pass filter

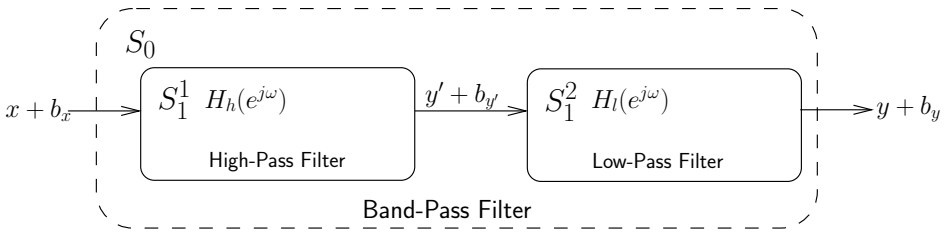


Figure 3.24: Hierarchical Decomposition of a Band-pass Filter

Consider a band-pass filter implemented using a cascade of high-pass and low-pass FIR[1] filters as shown in Figure 3.24. Hierarchically, the system consists of two sub-systems corresponding to the two filters. Both sub-systems S_1^1 (a high-pass filter) and S_1^2 (a low-pass filter) have memory and are hence frequency selective.

In the case of the sub-system S_1^1, the spectral characteristics of the output noise $b_{y'}$ is dissimilar from that of the input quantization noise b_x and has a particular power spectral density function shaped by the filter H_h. Similarly, in the case of sub-system S_1^2, the power spectral density of the input quantization noise $b_{y'}$ is shaped by the filter H_l. Hence, it is important to consider the frequency characteristics of both the sub-systems. The absence of un-smooth operators in the systems makes it unnecessary to derive the PDF parameters of the quantization noise.

The power spectral density of the output noise of both sub-systems S_1^1 and S_1^2 are defined using *Transmission filters:* $H_h^{tf}(e^{j\omega})$, $H_l^{tf}(e^{j\omega})$ and the *Generation filters:* $H_h^{gf}(e^{j\omega})$, $H_l^{gf}(e^{j\omega})$ respectively. Assuming a simple tapped-delay line implementation of the FIR filters; while the *Transmission filters* are obtained as the magnitude response of the filters. The transmission and generation filters are analytically derived as

[1]Finite Impulse Response

$$
\begin{aligned}
H_h^{tf}(e^{j\omega}) &= |H_h(e^{j\omega})|^2 \\
H_l^{tf}(e^{j\omega}) &= |H_l(e^{j\omega})|^2,
\end{aligned} \tag{3.41}
$$

where $\hat{H}_h(e^{j\omega})$ and $\hat{H}_l(e^{j\omega})$ in this case are the magnitude spectra obtained by evaluating transfer function $H_h(e^{j\omega})$ and $H_l(e^{j\omega})$ respectively. The noise generated due to quantization in the adders and multipliers pass through to the output without any attenuation or delay. The spectral parameters participating in the quantization noise generation and propagation for this example can be visualized as shown in the Figure 3.25.

$$
\begin{aligned}
H_h^{gf}(e^{j\omega}) &= 1 \\
H_l^{gf}(e^{j\omega}) &= 1.
\end{aligned} \tag{3.42}
$$

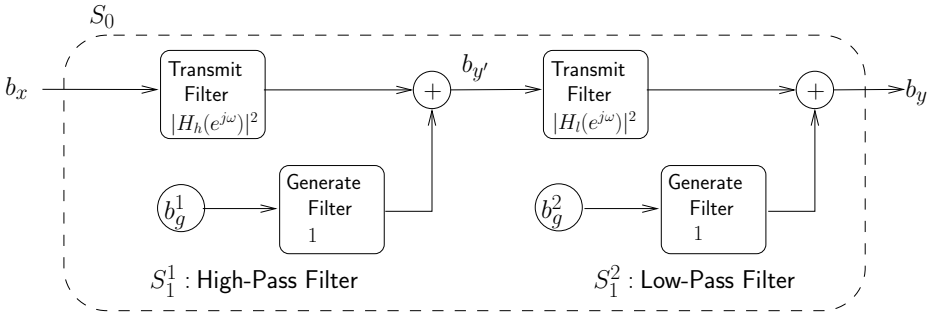


Figure 3.25: Single Noise Source components of the bandpass filter

The PSD of the output signals in each of the sub-system blocks can be calculated using the transmit and generate filters defined in Equations 3.41 and 3.42 as

$$
S_{b_{y'}b_{y'}} = H_h^{tf}(e^{j\omega}).S_{b_xb_x}(e^{j\omega}) + H_h^{gf}(e^{j\omega}).S_{b_{g1}b_{g1}}(e^{j\omega}) \tag{3.43}
$$

$$
S_{b_yb_y} = H_l^{tf}(e^{j\omega}).S_{b_{y'}b_{y'}}(e^{j\omega}) + H_lgf(e^{j\omega}).S_{b_{g2}b_{g2}}(e^{j\omega}), \tag{3.44}
$$

where $S_{b_{g1}b_{g1}}(e^{j\omega})$ and $S_{b_{g2}b_{g2}}(e^{j\omega})$ are the PSD of the total quantization noise generated within the sub-system S_1^1 and S_1^2 respectively. While the SNS model of the individual sub-systems can be calculated independently, the evaluation of noise is performed sequentially respecting the topological dependencies. In a feed-forward fashion, the output of the first sub-system is calculated before calculating the noise power at the output of the second sub-system.

The PSD information is recorded in the frequency domain by using discrete Fourier transform (DFT) coefficients. Representation using the DFT coefficients makes it easy to carry out filtering. The number of DFT points are defined by the user such that all the nuances of the frequency selectivities of the sub-systems are captured. Hence,

as discussed in Section 3.3.4, choosing greater number of FFT points will make the spectral representation more accurate at the expense of increased computational cost.

Filter with saturated arithmetic

The bits assigned for the integer part of the fixed-point format limits the dynamic range of the signal. In the saturation mode, the signal is clipped to the maximum and minimum values. Clearly, as long as the signal is within the saturation boundaries, the quantization error value propagates to the output with no modification. When the signal value crosses the saturation boundary, the error due to clipping of the waveform makes it dependent on the signal. The saturation operator is thus classified as an un-smooth operator.

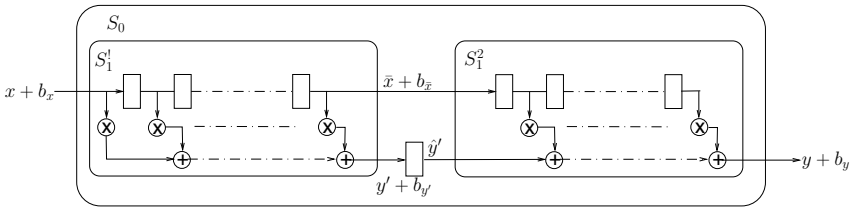


Figure 3.26: FIR with a saturation operator

Theoretically, saturation could occur soon after every adder in the filter data-path. Practically, it is impossible to check for saturation at the output of every operator. For the purpose of illustration, one of the filter sections is chosen and checked for saturation errors. Consider the FIR filter structure shown which includes a *saturation* operator as shown in Figure 3.26. The saturation operator sets limits on the maximum and minimum values a signal can take. These values can be called saturation boundaries. If the signal in infinite precision is already taking this value, the perturbation due to quantization will cause saturation of the signal. That is, the error value of the signal after saturation depends on the signal value and the noise power. The relationship between the actual error and the quantization noise power is not linear. This is referred to as un-smooth behavior. The lack of analytical techniques to model the un-smoothness in the saturation operator results in severing of the FIR filter into two sub-systems not including the saturation operator.

The first sub-system is a truncated FIR filter with M memory elements and its SNS model can be derived as it was done in the previous example. The quantization noise power spectrum at the output $b_{y'}$ and $b_{\bar{x}}$is given as

$$S_{b_{\bar{x}}b_{\bar{x}}} = e^{-j\omega M} S_{b_x b_x} \tag{3.45}$$

$$S_{b_{y'}b_{y'}} = |H_1(e^{j\omega})|^2 S_{b_x b_x}. \tag{3.46}$$

Here, $H_1(e^{j\omega})$ is the transfer function corresponding to the truncated filter with M memory elements and $S_{b_x b_x}$ is the PSD of the input signal.

It can be easily seen that the two outputs are correlated with one another. The error $b_{\bar{x}}$ is a delayed version of the input quantization error signal b_x. The cross-correlation between $b_{\bar{x}}$ and $b_{y'}$ is calculated and it can be written analytically as

$$S_{b_{\bar{x}}b_{y'}} = e^{-j\omega M} H_1^*(e^{j\omega}) S_{b_x b_x}(e^{j\omega}). \tag{3.47}$$

The second sub-system consists of $N - M$ memory elements with two inputs $\bar{x}$ and y' and one output y. Let $H_2(e^{j\omega})$ be the transfer function between one of its inputs $\bar{x}$ and the output y. The transfer function between the second input y' and the output is 1 as it does not involve any memory elements or scaling.

In the absence of saturation errors, the output PSD of the quantization noise at the output of the second system can be calculated analytically. The PSD of the quantization error signal b_y at the output of sub-system S_1^2 is given as

$$S_{b_y b_y}(e^{j\omega}) = S_{b_{y'}b_{y'}} + |H_2(e^{j\omega})|^2 S_{b_{\bar{x}}b_{\bar{x}}} + H_2^*(e^{-j\omega}) S_{b_{y'}b_{\bar{x}}}(e^{j\omega}) + H_2(e^{j\omega}) S_{b_{\bar{x}}b_{y'}}. \tag{3.48}$$

However, the lack of analytical models to capture the effects of saturation errors makes it imperative to use simulation for evaluation of fixed-point effects in cases where un-smooth operators are used. A hybrid approach which utilizes both partial analytical models along with fixed-point simulation is described in Chapter 4.

3.6 Summary

In this chapter, the problem of hierarchical decomposition of accuracy evaluation of large signal processing systems implemented using fixed-point operators is considered. The need for a hierarchical method is driven by the fact that the existing analytical methods have scalability issues with systems having a large number of operator noise sources. The hierarchical decomposition of the system into various sub-system hierarchies makes it possible to adopt a divide and conquer strategy to tackle the problem of fixed-point performance evaluation.

The framework of the Single Noise Source (SNS) model is then introduced to address the problem of accuracy evaluation of a hierarchically defined system as a function of noise-power at the output of its sub-systems. Qualitatively, this model is an extension to the idea behind the pseudo quantization noise (PQN) model which uses additive noise in order to introduce quantization errors instead of simulation with fixed-point simulation. The SNS model is also used similarly by using the proposed *evaluate-propagate-add* technique. However, using the PQN model is quite straight forward owing to its uniform distribution and white PSD characteristics. In case of the SNS model, the spectral and the distribution characteristics of the additive noise has to be derived from the respective sub-system parameters.

Analytical techniques to evaluate the auto-correlation and cross-correlation power spectrum and the probability distribution function of the total quantization noise generated within the given sub-system are discussed. The complexity of deriving these

characteristics are also discussed. Depending upon the topology of the interconnect between the sub-systems and the sub-system functionalities, it can be computationally difficult to evaluate all the SNS parameters. In order to reduce this overhead, an algorithm for selective evaluation of the SNS model parameters is also proposed. By using this procedure, it is assumed that output quantization noise has white spectral power distribution, uncorrelated with other noise signals and has Gaussian PDF characteristics unless otherwise mentioned.

Using the parameters specified by the SNS model, it should be possible to generate particular random processes such that it is statistically same as the quantization errors generated by simulating a fixed-point implementation of the system. Usually, owing to the large number of quantization noise sources it is expected that the quantization noise PDF is a Gaussian. Under such circumstances, it is quite easy to generate the particular random process with a specified spectral characteristics. On the other hand, the PDF shape can be ignored when there are no un-smooth operators. In this thesis, the focus is on proving the utility of the newly conceived SNS model. Actual generation of a random process mimicking the quantization noise is not trivial when the random process has a specified noise spectrum and non-Gaussian PDF. Under such circumstances, some of the techniques discussed in works such as [60] must be used. Exploration of techniques for generation of particular random processes is out of scope of this work.

Chapter 4

Un-smooth Operators

The use of fixed-point arithmetic operations leads to errors in computation with reference to the results obtained by using infinite precision operations. Quantization errors whose characteristics cannot be captured by the application of Pseudo Quantization Noise (PQN) model discussed in Chapter 2 are classified as un-smooth quantization errors. The PQN model works well when the step-size is comparatively smaller than the overall dynamic range of the signal as in the case of fixed-point arithmetic operations. In case of un-smooth quantization, the quantization step-sizes are very large and often comparable to the dynamic range of the signal being quantized. This violates the assumptions made for deriving the PQN model and hence the estimates obtained by using the PQN model can be very far from reality.

Any analytical model attempting to predict the error statistics at the output of an un-smooth quantization requires the knowledge of input signal and associated quantization noise characteristics. Moreover, in the presence of many un-smooth quantizers, the quantization error statistics are not only correlated with the signal but also with errors due to previous un-smooth quantizers. The lack of analytical techniques for noise propagation across un-smooth operations is one of the constraints on the seamless application of the Single Noise Source model described in Chapter 3.

In this chapter, the dynamics of quantization is re-looked at and an attempt to formally define the un-smooth quantizer is made in Section 4.1. With this definition, a technique to identify un-smooth quantizers in the context of the given signal and system characteristics is also proposed in Section 4.2. An analytical technique to estimate the error and the corresponding noise PDF at the output of an un-smooth quantizer which takes into account both: the signal statistics and the accumulated quantization noise statistics is devised in Section 4.5.

The un-smooth quantizer is just one type of un-smooth operation. The presence of other operations such as *min()*, *max()*, etc. in the system continue to challenge the ability to generate complete analytical expression for quantifying the impact of using fixed-point operators. For such cases, a hybrid technique to perform fixed-point performance analysis in the presence of un-smooth operators is proposed in Section 4.6. At the heart of this technique is the use of the Single Noise Source (SNS) model for

analytical evaluation of quantization errors due to smooth quantizers. This technique essentially uses the analytical techniques wherever possible in order to accelerate fixed-point simulation.

4.1 Dynamics of Uniform Quantization

The process of quantization is by definition un-smooth in the strict sense. The discontinuities at the boundary of every quantization bin is the source of un-smoothness. To determine the applicability of the PQN model to study the process of quantization, the relationship between the dynamic range α of the input signal and the quantization step-size q needs to be considered. When the quantization step size is much smaller than the dynamic range of the signal (i.e. $q << \alpha$), it is possible to model the errors introduced due to quantization using the PQN model. The small relative size is responsible for all the properties of the additive PQN model for quantization errors.

In case of un-smooth operators, the quantization step q is large and is often comparable to the dynamic range of the signal α. Under these circumstances, the properties of quantization error including the noise shape, its additivity property and its independence with input signal statistics are questionable. In general, these aspects of the quantization noise are known to vary with changing input signal PDF and also with quantization step-size.

In this section, the dynamics of quantization of real signals in the characteristic function domain is briefly recalled. This analysis is then applied to quantized signals to appreciate the quantization process during repeated quantization.

4.1.1 Quantization in Characteristic Function Domain

The phenomena of quantization is similar to the Nyquist sampling. While the Nyquist sampling is defined along the time axis, quantization can be thought of as sampling along the amplitude axis of the signal. Thereby, the PDF of the signal and its characteristic function (CF) are analogous to the time domain signal and its Fourier domain representation. Carrying the same analogy further, the quantization step q is similar to the Nyquist sampling period T.

The CF $\Phi_x(u)$ of the original signal x repeats itself along the CF domain axis u, the quantization radian frequency with the interval $\Psi = \frac{2\pi}{q}$. Figure 4.1 shows aliasing of the characteristic functions of the quantized signal when quantized with step-size is greater than or equal to q_b. The quantization bandwidth of the signal which is being quantized is defined as the quantization radian frequency $\frac{2\pi}{q_b}$ corresponding to the step-size q_b at which aliasing starts to occur between successive characteristic function images. The quantization bandwidth W_x of a signal x is a function of the quantization step size and the characteristic function of the signal being quantized. It can be written as

$$W_x = \frac{2\pi}{q_b} \quad \text{such that} \quad \Phi_x(u) = 0 \quad \forall \quad u \geq \frac{2\pi}{q_b}. \tag{4.1}$$

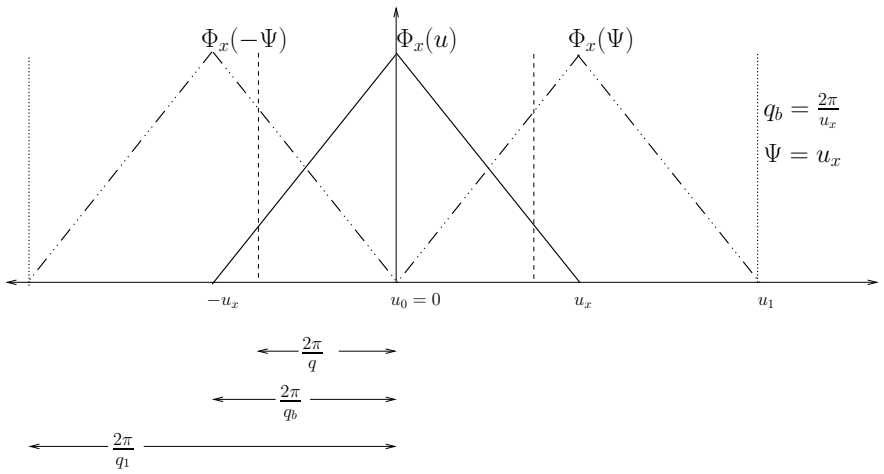


Figure 4.1: Quantization theorems 1 & 2: characteristic function domain

When the quantization step is small enough, the radian frequency is large not to allow overlapping of the $\Phi_x(u)$. As the quantization step size increases, the repetition is more frequent causing aliasing and hence the recovery of the original PDF from the CF becomes impossible. This observation is referred to as the first quantization theorem (QT1). In other words, this is nothing but the application of Nyquist sampling theorem along the amplitude axis of the signal. In Figure 4.1, q_1 is the largest quantization step-size such that QT1 is satisfied.

In the study of quantization effects, correctness of the PQN model is of interest rather than the exact recovery of the continuous signal. The second quantization theorem (QT2) relaxes the conditions set by QT1 further and allows overlap of the repeating images of the CF of the signal. The second quantization theorem (QT2) states the conditions under which the moments of the original signal can be recovered from the quantized signal. In Figure 4.1, the largest quantization step-size that can satisfy the condition in QT2 is denoted as q_b.

The moments of the quantized signal $\dot{x}$ is given by

$$E\{\dot{x}^r\} = E\{x^r\} + S_r + R_r, \tag{4.2}$$

where S_r is the r^{th} moment of the quantization error signal and R_r corresponds to the covariance between the r^{th} moment of the input signal x and the quantization error. When QT1 or QT2 are satisfied, the value of the residue coefficient R_r takes a value of 0 for all r. In other words, the PQN model can be used to analytically determine the effect of quantization noise. The quantization step-size as large as q_b marks the boundary between the applicability and non-applicability of the PQN model. When the quantization step-size is increased beyond this boundary, the value of R_r becomes non-zero and results in deviation from the PQN model.

In practice, the input signal assumes a finite set of values and is hence limited in the PDF domain. Theoretically, this renders the CF of the signal band unlimited. In

other words, unlike the case depicted in Figure 4.1, the CF actually is theoretically zero only at infinity. Quantization theorems QT3 and QT4 pitches in with conditions when any arbitrary moment of the quantization noise and the signal can be mutually orthogonal (i.e. R^r in Equation 4.2 is 0 for any r) and the shape of the quantization noise continues to be uniformly distributed irrespective of where the CF goes to 0. Accordingly, the PQN model is applicable on a quantizer with step-size q if the input characteristic function and its derivatives take up the value of 0 at all points corresponding to the multiples of the quantization radian frequency: $\Psi = \frac{2\pi}{q}$. A formal and rigorous description of the quantization theorems and their corollaries are described in [120].

4.1.2 Dynamics of Double Quantization

Any system in practice consists of many quantizers that cause repeated quantization along a fixed-point data path. While the quantization theorems are well defined for the first quantization, the applicability of the PQN model for the double quantization phenomena is not rigorously defined. In Chapter 3, it was assumed that all quantizers comply with the conditions for PQN always. In order to justify that assumption, consider the double quantization as shown in Figure 4.2. Without loss of generality, simple rounding mode is considered for both quantizers. Although the total noise power is affected due to a finite non-zero mean, the dynamics of the quantization is not impacted in the truncation modes.

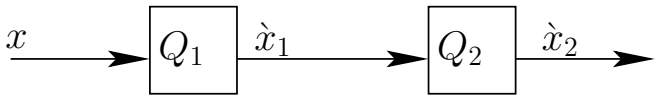


Figure 4.2: Double quantization

In [120], the PDF of the quantized signal at the output of the first quantizer Q_{K1}: $f_{\grave{x}_1}(x)$ is obtained by the area sampling process as

$$\begin{aligned} f_{\grave{x}_1}(x) &= (f_x(x) * f_{n_1}(x)) \cdot c_{q_1}(x) \\ &= \sum_{m=-\infty}^{\infty} \delta(x - m \cdot q_1) \int_{m \cdot q_1 - \frac{q_1}{2}}^{m \cdot q_1 + \frac{q_1}{2}} f_x(\alpha) d\alpha, \end{aligned} \tag{4.3}$$

where δ is the impulse function, $f_{x_k}(x)$ and $f_{n_1}(x)$ are respectively the PDFs of input signal x and the quantization noise at the first quantizer Q_1. $c_{q_1}(x)$ is the train of impulses corresponding to the quantizer Q_1 spaced q_1 units apart given as

$$c_{q_1}(x) = \sum_{m=-\infty}^{\infty} q_1 \cdot \delta(x - m \cdot q_1). \tag{4.4}$$

The second quantizer input is the signal $\grave{x}_1$, which is discrete in nature. However, the dynamics of quantization of discrete PDF signals is unchanged. Due to the discrete amplitude of the input signal, it is sufficient to consider the error due to quantization at the discrete points. The discrete PDF[1] of the quantization noise $f_{n_2}(x)$ can be written by further discretizing the range with the quantization step size of the previous quantizer q_1 as

$$f_{n_2}(p) = \begin{cases} \frac{1}{k} & -\frac{k}{2} \leq p < \frac{k}{2} \\ 0 & \text{elsewhere} \end{cases} \tag{4.5}$$

where $p = \frac{x}{q_1}$ is the discrete variable corresponding to x and k is the ratio of quantization step-sizes $k = \frac{q_2}{q_1}$. When binary arithmetic circuits are used, it has to be noted that the step-sizes of the successive quantizations are multiples of powers of 2. In other words, sampling the impulse train of the quantizer $c_{q_1}(x)$ at the rate of k gives $c_{q_2}(x)$. Its discretized version can be written as

$$c_{q_2}(p) = \sum_{m=-\infty}^{\infty} k \cdot \delta(p - m \cdot k). \tag{4.6}$$

The PDF of the signal $\grave{x}_2$ can be constructed as in Equation 4.3 and can be written as

$$\begin{aligned} f_{\grave{x}_2}(p) &= (f_{\grave{x}_1}(p) * f_{n_2}(p)) \cdot c_{q_2}(p). \\ &= \left(\sum_{\beta=-\infty}^{\infty} f_{n_2}(p-\beta) \cdot f_{\grave{x}_1}(\beta) \right) \cdot c_{q_2}(p). \\ &= \left(\sum_{\beta=p-\frac{k}{2}}^{p+\frac{k}{2}} \frac{1}{k} \cdot f_{\grave{x}_1}(\beta) \right) \cdot c_{q_2}(p). \\ &= \left(\sum_{\beta=p-\frac{k}{2}}^{p+\frac{k}{2}} \frac{1}{k} \cdot f_{\grave{x}_1}(\beta) \right) \cdot \sum_{m=-\infty}^{\infty} k \cdot \delta(p - m \cdot k). \\ &= \sum_{m=-\infty}^{\infty} \delta(p - m \cdot k) \sum_{\beta=m \cdot k-\frac{k}{2}}^{m \cdot k+\frac{k}{2}} f_{\grave{x}_1}(\beta). \end{aligned} \tag{4.7}$$

The PDF of the signal x_2 as a function of the discrete PDF $f_{\grave{x}_1}(p)$ where β counts k discrete samples of PDF of signal $\grave{x}_1$ centred around $m \cdot k$ for all m. It can be simplified

[1] Discrete PDF is also called as probability mass function (PMF).

as a function of the signal in infinite precision as

$$
\begin{aligned}
f_{\dot{x}_2}(\frac{x}{q_1}) &= \sum_{m=-\infty}^{\infty} \delta(\frac{x}{q_1} - m\cdot\frac{q_2}{q_1}) \cdot \sum_{\frac{x}{q_1}=m\cdot\frac{q_2}{q_1}-\frac{q_2}{2q_1}}^{m\cdot\frac{q2}{q1}+\frac{q_2}{2q_1}} f_{\dot{x_1}}(\frac{x}{q_1}). \\
f_{\dot{x}_2}(x) &= \sum_{m=-\infty}^{\infty} \delta(x - m\cdot q_2) \sum_{x=m\cdot q_2-\frac{q_2}{2}}^{m\cdot q_2+\frac{q_2}{2}} f_{\dot{x_1}}(x). \qquad (4.8)
\end{aligned}
$$

Now, consider the case when a single quantizer Q_2 with step size $q_2 = k.q_1$ is used to obtain $\dot{x}_2$. The PDF of the signal directly can be obtained directly from Equation 4.3 as

$$
f_{\dot{x}_{2'}}(x) = \sum_{m=-\infty}^{\infty} \delta(x - mq_2) \int_{mq_2-\frac{q_2}{2}}^{mq_2+\frac{q_2}{2}} f_x(\alpha)d\alpha. \qquad (4.9)
$$

The quantization step-size of quantizer Q_2 is an integer multiple of the step-size of the first quantizer Q_1. Hence, the total area covered by the integral between the ranges $(-\frac{q_2}{2}, \frac{q_2}{2})$ is as good as adding up k samples of the PDF $f_{\dot{x_1}}(x)$ in the range. That is,

$$
\int_{mq_2-\frac{q_2}{2}}^{mq_2+\frac{q_2}{2}} f_x(\alpha)d\alpha = \sum_{x=m\cdot q_2-\frac{q_2}{2}}^{m\cdot q_2+\frac{q_2}{2}} f_{\dot{x_1}}(x). \qquad (4.10)
$$

By comparing Equations 4.9 and 4.8, it is clear that the PDF of the quantized signal in case of double quantization is the same as it is in the case of single quantization.

This result means that the quantization theorems are applicable even in the double quantization case as if it is being applied on the original signal. In other words, the PQN model can be applied to estimate the total quantization noise statistics at the output of any quantizer irrespective of whether the quantizer is acting on a continuous signal or an already quantized signal.

4.2 Defining Un-smooth Quantization

In the light of the characteristic function domain behavior of quantizers, it is easy to reason out why the PQN model fails when the quantization step-size is large and comparable to the dynamic range. In general, as the quantization step size increases, the separation between the various images of the characteristic function as shown in Figure 4.1 decreases. In practice, the signal PDF is limited in its range and hence the unlimited in its characteristic domain representation. Therefore, the aliasing between successive images in the characteristic function domain occurs at even small step-sizes. However, the magnitude of aliasing remains small and hence the estimates of moments

obtained by the PQN model is not very far from the actual errors. However, the aliasing increases with growing step size also grows and the estimates as described by the PQN statistics starts deviating from the observed error statistics significantly. The un-smooth quantization can therefore be described as

Definition: A quantizer is *un-smooth* if the deviation of its actual error statistics is unacceptably large from the error statistics as determined by the PQN model.

It has to be noted that unless the characteristic function is band-limited, the error in estimation is always present. The magnitude of this error increases with increasing step-sizes. The acceptability of the error is therefore a user defined parameter. The trade-off here is that if a large deviation is acceptable, many quantizers in the system can be regarded as smooth. On the other hand, being very strict about the errors makes most quantizers un-smooth.

Given that one of the quantizers in the system is considered smooth, the smoothness of other quantizers can be deduced with the knowledge of the bandwidth of the signal in its characteristic function domain.

Theorem: If a quantizer Q_{ref} with step-size q_{ref} quantizing the signal ω_{ref} with quantization-bandwidth W_{ref} as defined in Equation 4.1 is considered smooth, any other quantizer Q_k with step size q_k quantizing the signal ω_k whose quantization-bandwidth is also smooth if $\frac{q_{ref}}{W_{ref}} \geq \frac{q_k}{W_k}$.

Proof: The quantization bandwidth defines the spread of the characteristic function and the quantization step-size determines the rate at which the replicas of the characteristic functions are found along the frequency axis. The reference quantizer Q_{ref} is given as smooth. This essentially means that the replicas of the characteristic function are separated enough not to cause significant aliasing.
If the quantizer Q_{ref} is used to quantize a signal ω_k whose quantization-bandwidth $W_k \leq W_{ref}$, the replicas of the characteristic function are relatively spaced farther apart than in the given reference case. Therefore, it continues to be smooth. Further, if the quantization step-size of the quantizer is further reduced to q_k such that $q_k < q_{ref}$, the replicas of the characteristic function are separated even further.
Thus, with reference to the given smooth quantizer it is sufficient to check for the conditions given in the theorem statement to deduce smoothness of other quantizers with respect to a given smooth quantizer for reference.

It has to be noted that this theorem provides sufficient conditions and it is not the necessary condition. That is even when $W_k > W_{ref}$, it is possible that the quantization is smooth. In such cases, the definition of un-smoothness has to be checked.

4.2.1 DCT Based Image Compression

To illustrate the effects of un-smoothness, an experiment which demonstrates the effect of error between the estimates suggested by the analytical PQN model and actual quantization error is presented. This experiment is derived by reducing the process of DCT-based[1] image compression. The experimental setup is as shown in the block diagram in Figure 4.3.

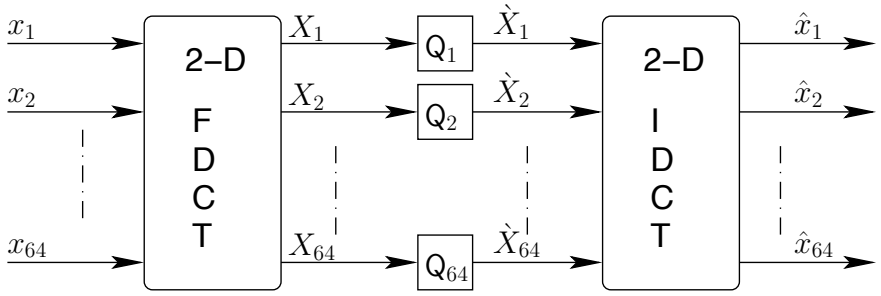


Figure 4.3: Quantization in JPEG

According to the JPEG image compression standard, in the image encoding phase, forward discrete cosine transform (FDCT) is performed on 8×8 blocks of pixels corresponding to the vector $\mathbf{x} = \{x_1, x_2, ...x_{64}\}$ to obtain the transform coefficients $\mathbf{X} = \{X_1, X_2...X_{64}\}$. These coefficients are scaled down by the respective quantizer step-sizes as given by the quantization matrix specified in the JPEG standard. These scaled down coefficients are encoded using Huffman codes and packaged into a compressed file. In the image decoding phase, the Huffman codes are decoded to obtain scaled down transform coefficients. These coefficients are scaled up by the respective scaling factor to obtain $\grave{\mathbf{X}} = \{\grave{X}_1, \grave{X}_2...\grave{X}_{64}\}$ and the inverse discrete cosine transform (IDCT) is applied to reconstruct the original image.

The up-scaling and down-scaling of DCT coefficients correspond to rounding quantization whose step-size corresponds to the coefficients of the quantization matrix as described in the JPEG standard. That is, the k^{th} coefficient X_k is quantized by a quantizer Q_k with step-size q_k to obtain $\grave{X}_k$ as shown in Figure 4.3. The image compression standard exploits the fact that natural images tend to be band-limited for specifying the value of quantization step size. Therefore, the quantization step-sizes are smaller for low frequency components and larger for high frequency components. That is $q_{64} > q_{63} > \ldots > q_2 > q_1$.

This experiment measures the effects of quantization of transform coefficients on quality of the image. The FDCT and IDCT makes use of double precision arithmetic and hence the errors introduced during transforms computation are negligible. The later stages involving entropy coding does not introduce any loss of information.

In this experimental setup, the input vector $\mathbf{x}$ is derived from a Gaussian random number generator. Consequently, the transform coefficients also have Gaussian distri-

[1] JPEG, MPEG-1,2,4 are popularly used DCT-based image compression standards

bution with the same signal power. Amongst these 64 quantizers, some of them exhibit smooth quantization while others exhibit un-smooth properties.

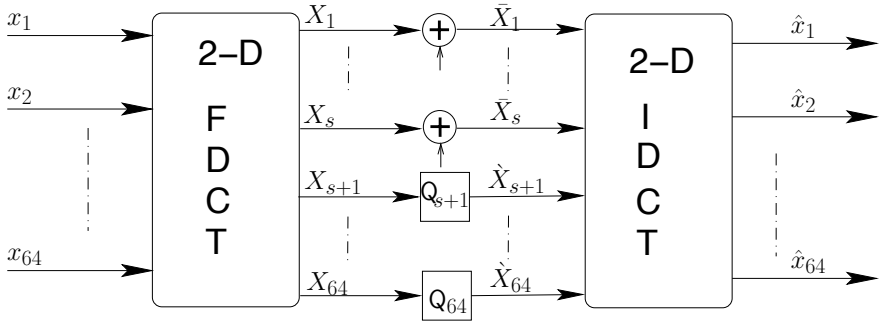


Figure 4.4: Application of PQN model in the place of quantization in JPEG

Consider the application of the PQN model for all the quantizers. According to the PQN model, a uniform noise source b_k corresponding to quantization step-size q_k in rounding mode has a power of $\frac{{q_k}^2}{12}$ with zero mean. An additive noise source corresponding to these characteristics is statistically equivalent to actually performing the quantization Q_k. If a quantizer Q_k is considered smooth, a random value from the uniform noise source b_k is generated and added to X_k to obtain $\bar{X}_k$. If it is considered un-smooth, the quantization operation corresponding to Q_k is actually performed.

This is applied to the example under consideration as shown in Figure 4.4. Depending upon the tolerance of error, some of the quantizers can be considered smooth and an additive random noise with appropriate noise-power is used in place of the quantizer. The boundary between smooth and un-smooth quantization is marked by the change in using the real quantizer instead of the estimates derived from the PQN model. In the Figure 4.4, this boundary is defined such that quantizer with step-size q_s is smooth. This corresponds to the s^{th} coefficient. The statistics of all the transform coefficients have a Gaussian distribution and have the same power. Also, the quantization step-size for coefficients whose index is less than s have smaller quantization step-sizes. Therefore, all quantizers whose index is smaller than s are smooth and quantizers Q_1 through Q_s can be approximated by additive quantization noise whereas the quantizers q_{s+1} through Q_{64} continue are preserved for the purpose of performing quantization operation.

If the quantizer under consideration is indeed smooth, adding the noise generated from the random noise source b_k will not affect the reconstruction of the image. In other words, the error between the reconstructed image obtained by actually performing the quantization step and by adding quantization noise b_k will be statistically equivalent. In cases where quantizers though un-smooth are replaced by the PQN noise source, the error statistics between the original and the reconstructed image deviates largely from the actual error due to quantization.

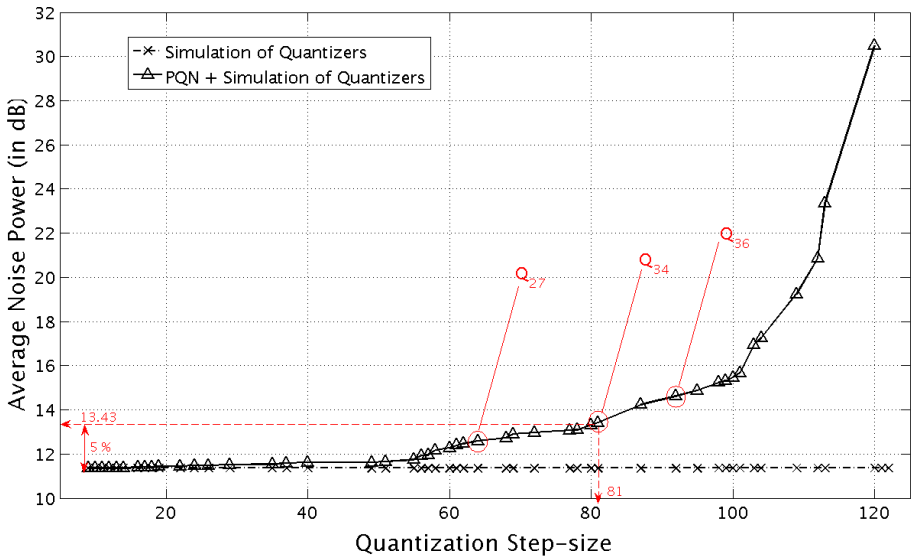


Figure 4.5: Quantization in JPEG

4.2.2 Defining the Un-smooth Boundary

The graph in Figure 4.5 shows the increase in average noise power (in decibels) as the quantizers starting from the smallest step size is considered smooth. That is, all quantizers are considered smooth starting from the quantizer with the smallest step size in ascending order. Thereby, the experiment is carried out by considering all the quantizers to be smooth eventually. In other words, the smoothness constraint is relaxed to allow more and more deviation of error statistics from the PQN model. In every step, one more quantizer is replaced by the corresponding PQN noise model. The results obtained is compared with the result obtained by simulation.

Here, the signal power of input to each transform coefficient is the same and hence normalization of input signal power is not necessary to study the impact of quantization step-size. To begin with, the quantization step sizes are small enough not to cause a large error between the original and the reconstructed image. As the smoothness constraint becomes more and more lenient, the gap between the average noise obtained by the selective use of PQN model on small quantizers begin to widen even as all the quantizers are considered smooth.

Clearly, depending upon the users tolerance to error in estimation, it is possible to define a quantization step-size by following the curve obtained in Figure 4.5 beyond which the estimation of error could be unacceptable. For example, in the case of this experiment with DCT coefficients, if about 5% relative error in estimation is tolerated by the application, the corresponding quantization step-size is 81 units. This step-size corresponds to the quantizer Q_{34}. From the graph in Figure 4.5, if any quantizer with index lesser than 34 such as quantizer Q_{27} is chosen, the relative error in estimation of

quantization noise decreases. Whereas if any other quantizer with index greater than 3 such as the quantizer Q_{36} is chosen, the relative error would only increase.

4.3 Identifying Un-Smooth Operators

In a practical scenario, a fixed-point system consists of a number of quantization operations. The analytical estimates obtained by the use of PQN model can be erroneous when it is applied on un-smooth quantizers. Therefore, it is required to classify the quantizers as smooth or un-smooth based upon the characteristics of the signal being quantized and the quantization step-size.

The Algorithm 4.1 is based on computation of the characteristic function and proposes a technique to check for the applicability of the PQN model in case of any uniform quantization operator. This algorithm identifies the boundaries of quantization step-size such that the PQN model can be applied to model the error behavior.

Algorithm 4.1 :: Identify Un-Smooth Boundaries

1: $G(V, E)$ = GetSystemGraph();
2: **for all** Q_i quantizers in $G(V, E)$ **do**
3: Obtain experimental PDF $f_{x_i}(x_i)$;
4: b = GetMaxBits(); * *Maximum number of bits assignable* *\
5: $\varepsilon_i^b = 0$
6: $\mathbf{Q_i^b} = b$; * *Assuming that the quantizer is smooth* *\
7: **while** $\varepsilon_i^b < \tau_i$ **do**
8: ε_i^b = EvaluateDeviation$(f_{x_i}, 2^{-b})$;
9: **if** $\varepsilon_i \geq \tau_i$ **then**
10: $\mathbf{Q_i^b} = b + 1$; * O_i *becomes un-smooth at* b *\
11: break;
12: **end if**
13: $b = b - 1$ * *Decrement the number of bits assigned* *\
14: **end while**
15: **end for**

The algorithm begins by considering all the quantization operations in the system graph $G(V, E)$ consisting of V operators interconnected by E edges. An operator is represented by one of the nodes with multiple input edges and a single output edge. Such an operation, when implemented using finite precision can essentially be represented using the schematic of the operator followed by the quantizer. The schematic of an operation in infinite precision and finite precision are as show in Figure 4.6.

In a fixed-point implementation, the inputs to the operators are also quantized. The effect of input quantization is covered by the quantizers present at the output of the operator nodes from which the signal emanates. Therefore, one quantizer can be associated with every signal. The effect of quantized inputs has an effect on the error behavior of the quantizer at the output of the given operator. In this scenario, it is

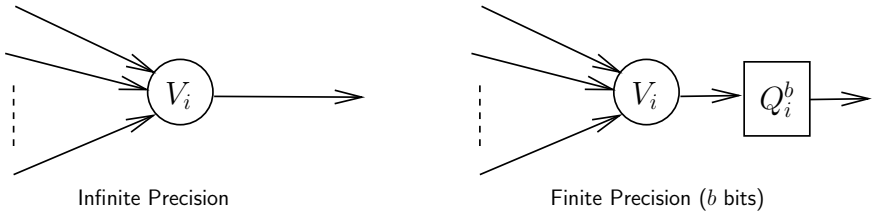


Figure 4.6: Operator O_i in infinite precision and finite precision with b-bits

possible to have double quantization effects. However, it is clear from Equations 4.8 and 4.9 that repeated quantization does not fundamentally change the quantization dynamics. Hence, to evaluate smoothness of each quantization, it is sufficient to work with the infinite precision data statistics at all points of interest. A technique to determine the same is presented in Algorithm 4.1.

The algorithm begins by considering all quantizers Q_i to be smooth. An infinite precision[1] simulation consisting of an exhaustive set of test vectors is used to capture the PDF characteristics: $f_{x_i}(x_i)$ of the signal at the input of the i^{th} quantizer Q_i. Alternatively, the signal PDF can also be estimated by using some of the techniques presented in Section 2.2.

The test for deviation in error begins with the assignment of maximum number of bits to the quantizer under consideration. The deviation ε_i^b of the i^{th} signal with b bits of quantization is evaluated either by simulation or analytically as described in [120]. The term τ_i is the tolerance to error in estimation which is defined by the user. This can be set specifically for a given quantizer or it can be universally set to one single value applicable to all quantizers in the system. While the deviation from PQN model is smaller than τ_i, the number of bits assigned continues to be decremented by 1 and the deviation from PQN model is re-evaluated for the newly assigned word-length. When the loop exits, the value of Q_b^i holds the value of b at which the quantization turns out to be un-smooth. Alternatively, a binary search procedure can be adopted instead of decrementing one bit at a time in order to reduce the number of times the loop is executed.

When the error deviation is larger than the tolerance, it means that the quantizer which was smooth until the previous quantization step trial is now un-smooth. In other words, this transition marks the boundary between smooth and un-smooth quantization step-sizes for the signal associated with the i^{th} signal. Therefore, Q_i^b is assigned the value of $(b+1)$ to mark the smallest word-length under which this quantizer can be considered smooth.

To understand the behavior of the deviation from PQN model, consider a signal normalized in the range $[-1, 1)$. Its fixed-point format is as given in Figure 4.7. Figure 4.8 shows the magnitude of relative error between the analytical PQN model and the error estimates obtained by simulation. For the sake of illustration, three relatively simple PDF (viz. Gaussian, Laplacian and uniform) are considered.

[1] Double precision floating point simulation is used as an approximation to infinite precision.

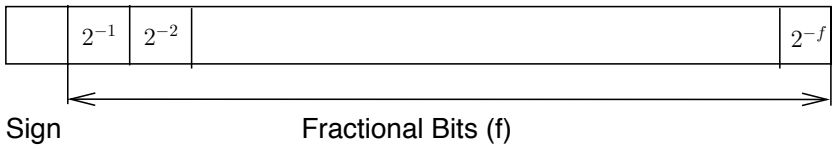


Figure 4.7: Fixed-point representation of signal with dynamic range $[-1, 1)$

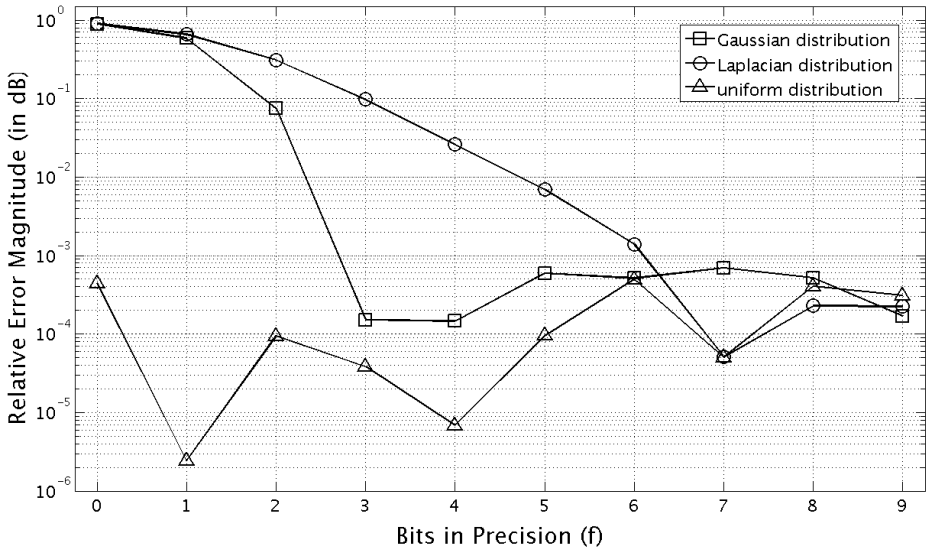


Figure 4.8: Relative Error Magnitude (in dB) between PQN and simulation

The relative error between PQN and simulation is the highest for the Laplacian, the deviation is a little less for the Gaussian and very small and negligible in case of the uniform signal distributions. The uniform distribution has values spread across the range $[-1, 1)$ and hence quantization with the largest number of bits preserves the uniform shapes and the value of -1 or 1 contributes to the total noise power. Whereas, in case of the Gaussian, the values are denser near the value of 0 and therefore it gets zeroed out when large quantization step-sizes are used. This results in large deviation between the PQN error estimates and the errors observed by simulation. This effect is more pronounced in case of the Laplacian as more values than in the case of the Gaussian are closer to 0. As the quantization step-size is reduced (i.e. more bits are assigned for precision), the values are not zeroed out and hence converge to similar behavior.

In the above algorithm, the statistics of the value taken by every signal associated with a quantization operator in the system needs to be recorded. Indeed, storing the values of all signals through the entire length of simulation can consume a lot of memory and time and may not be practically feasible. It is sufficient to capture the signal statistics of only those signals which are associated with quantizers suspected to

be un-smooth.

4.4 Un-smooth Quantization Errors

It is common to find smooth and un-smooth quantizers coexisting in a number of signal processing systems. In communication systems for example, they are most commonly found in receiver algorithms in the form of QAM decision operators and co-exist with other smooth fixed-point arithmetic operators. While the smooth quantization errors can be estimated using the additive PQN model, the response of QAM slicers to perturbation by quantization noise can be non-linear and therefore can cause large deviation from the estimates obtained by the application of the PQN model. Therefore, it is important to understand the genesis of errors due to un-smooth quantizers. In particular, since the smooth quantization operators co-exist with un-smooth quantization operators, estimating the error statistics at the output of an un-smooth quantization operator due to perturbation of the signal at its input becomes necessary.

4.4.1 The Decision Operator

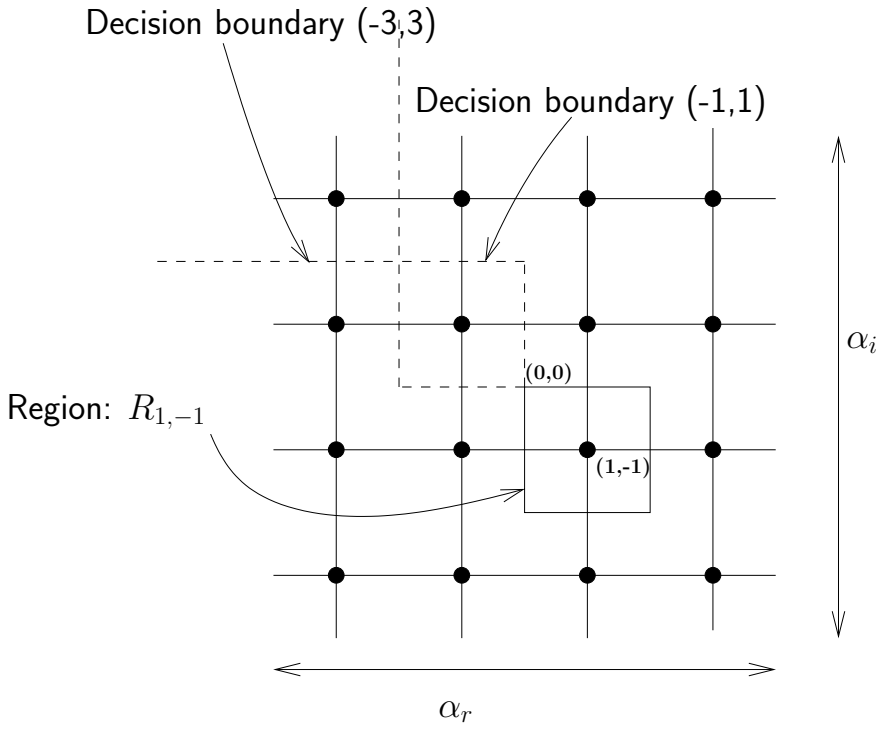


Figure 4.9: 16-QAM Constellation diagram

The decision operator or the slicer, used for discriminating between different values post equalization in a digital communication system is a very good example for an

un-smooth quantizer. The constellation diagram of a 16-QAM scheme is as shown in Figure 4.9. Behavior of a QAM slicer used to discriminate the values of a given signal x into the QAM constellation symbols given by the set $\mathbf{S} = \{S_1, S_2..., S_L\}, \forall S_i \in R^N$ (where N is the dimensionality of the signal transmitted[1]) is defined as

$$\tilde{x} = S_i \quad \text{if} \quad x \in R_i, \tag{4.11}$$

where R_i is the region defined as the neighbourhood of the constellation point S_i. The region R_i is a space in R^N and essentially consists of the set of values taken by x that can be discriminated as S_i. In Figure 4.9, the region $R_{1,-1}$ is the area covered by the rectangle around the constellation point $(1, -1)$. Likewise, the boundary of the QAM constellation symbol $(-1, 1)$ and $(-3, 3)$ is also defined by dashed lines. The area contained within these boundaries is referred to as the region $R_{-1,1}$ and $R_{-3,3}$ respectively.

Interestingly, the QAM decision operator bears a lot of similarities with the quantizer encountered in the study of finite precision arithmetic. The functioning of the QAM slicer is similar along both the real (along the horizontal directions) and imaginary (along the vertical directions) axes. The behavior of the QAM slicer along any of the axes can be thought of as comprising of a quantizer Q with step-size q followed by a saturation operator S with dynamic range α as shown in Figure 4.10. The quantizer Q is rounding in nature and centered around the constellation points. The dynamic range α is the maximum difference between any two constellation points. When the dynamic range of the input signal is comparable to the dynamic range of the saturation operator (i.e. $max(x) - min(x) \simeq \alpha$) the saturation operator may be ignored (i.e. $\tilde{x} \simeq \grave{x}$) and the decision operator can be approximated as a quantizer.

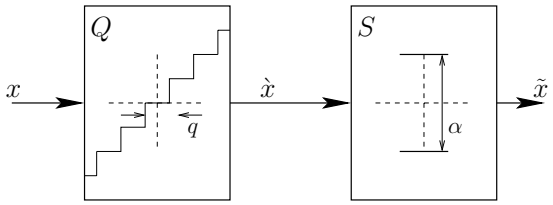


Figure 4.10: Quantization model of the Strong Decision Operator

While the behavior of the QAM decision operator is the same as in the case of a quantizer which is used to model the finite precision arithmetic, the error characteristics are not un-smooth. The difference between the QAM decision operator and the smooth quantizer that is usually encountered in finite precision implementation is that of the large relative step-size with respect to the signal. While the quantization step-size due to finite precision is usually orders of magnitude smaller than the dynamic range, the quantization step-size in the case of decision operator is often comparable to the signal itself. Therefore, the decision operator is un-smooth.

[1] $N = 2$ for QAM signals

4.4.2 Response to Perturbation

In the case of un-smooth operators, the output error statistics does not comply with the PQN model. Hence, the impact of quantization noise at the output of an un-smooth quantizer can be quantified only by comparing the response of the un-smooth operator to the input values in case of finite precision and its infinite precision counterparts. The effect of perturbation due to accumulated quantization noise at the input of a un-smooth quantizer is depicted in the Figure 4.11.

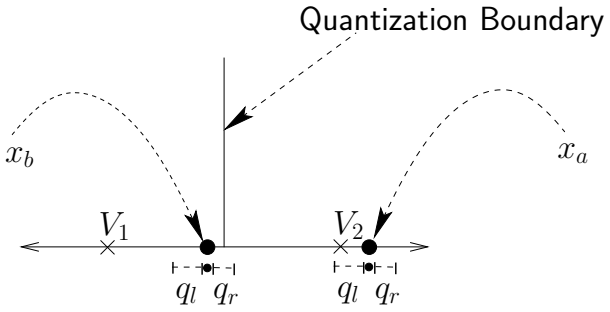


Figure 4.11: Response to perturbation at un-smooth boundary; Case A: $x = x_a$, Case B: $x = x_b$

A signal x is input to the un-smooth operator whose output is either the value V_1 or V_2 depending upon the value of x. Here, two scenarios in which a signal x assumes values x_a and x_b in infinite precision are depicted. In a fixed-point implementation, the signal x is perturbed due to accumulated quantization noise. Let q_l and q_r be the maximum probable negative and positive distortions. In cases when the signal assumes a value sufficiently far from the boundary as depicted in *Case A:* (i.e. $x = x_a$), the perturbation magnitude q_l or q_r is not large enough to make the signal cross the boundary. Hence, the value assigned by the un-smooth operator in both infinite precision and finite precision is V_2 and no error is propagated. On the other hand, if any double precision value x is close enough to the boundary as depicted in *Case B:* (i.e. $x = x_b$), a positive perturbation due to quantization error could push the actual fixed-point value across the boundary and cause the output to be V_2 instead of V_1 thus causing the fixed-point implementation behave differently from the infinite precision system behavior.

4.5 Computing Error Probability

The effect of quantization can be measured by defining a metric which compares the values of various signals in the system when fixed-point operators are used with the values assigned to them in the ideal case where infinite precision operators are used. In case of smooth quantizers, the errors are small and therefore conveniently captured by the quantization noise. In the case of un-smooth quantizers, the power of errors does not confirm with the analytical PQN error model. Therefore, it becomes important to obtain the probability density function of error at the output of the un-smooth

operator. Moreover, the error power can be derived very easily if the PDF of the error is available. Computing the error probability at the output of the quantizer is therefore more fundamental than directly deriving noise-power of the error signal due to quantization.

4.5.1 Decision Error Statistics

To study the impact of accumulated smooth quantization noise on the first decision output, a signal processing system which has one un-smooth operator U_1 at the output of the smooth system $\hat{x}$ is considered. Such scenarios commonly occur when considering a communication receiver systems as shown in Figure 4.12.

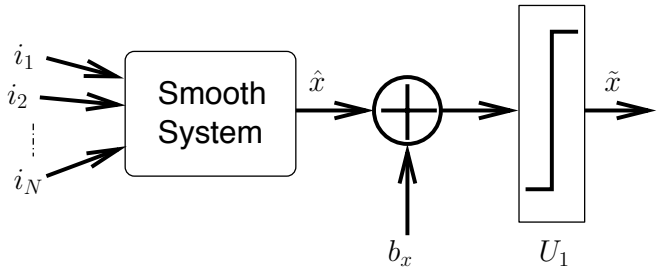


Figure 4.12: System with one un-smooth operator at the output

The value of any signal $\hat{x}_{fp}(n)$ at the output of the smooth system fixed-point block is obtained by perturbation of $\hat{x}_{ip}(n)$ due to quantization noise $b_x(n)$. Therefore, it can be written as

$$\hat{x}_{fp}(n) = \hat{x}_{ip}(n) + b_x(n). \tag{4.12}$$

The decision error rate due to additive quantization noise can be arrived at by comparing the fixed-point output $\tilde{x}_{fp}(n)$ with that of output obtained by system with infinite precision $\tilde{x}_{ip}(n)$.

As discussed in Section 4.4.2, the drift in signal value in a fixed-point system is a function of the quantization noise power associated with the signal. When the signal value is close to the decision boundary and the quantization noise power is sufficiently large, the drift caused due to the perturbation can be such that it crosses over the decision boundary generating different outputs in comparison with the output produced by a system with infinite precision implementation. If the quantization noise power is sufficiently large, there can also be chances where the drift in the signal value can cross multiple boundaries. This is perceived as a decision error at the output.

Let $P_{i,j}$ be the probability that the decisions $\tilde{x}_{ip} = S_i$ and $\tilde{x}_{fp} = S_j$ as defined in Equation 4.11. Consider a case when the input value which happens to be in the region R_i is perturbed by the noise b_x such that it now happens to be in a different boundary R_j.

$$P_{i,j} = \text{Prob}\{(\hat{x}_{ip} \in R_i) \cap (\hat{x}_{ip} + b_x \in R_j)\}. \tag{4.13}$$

Let $f_b(b)$ and $f_x(x)$ be probability density functions of the noise b_x and the infinite precision signal $\hat{x}_{ip}$ respectively. The probability P_i that the signal $\hat{x}_{ip}$ gives a decision $\tilde{x}_{ip} = S_i$ is obtained by integrating the PDF over region of interest R_i and is given by

$$P_i = \int_{R_i} f_x(x)dx. \tag{4.14}$$

Consider an instance when the signal x takes the value x_i such that $\tilde{x} = S_i$. That is, $x_i \in R_i$. With the knowledge of PDF of the quantization noise b_x, it is possible to calculate the error probability $P_{x_i,j}$ which is the error due to perturbation of the signal with value x_i which leads to symbol S_i in infinite precision but leads to symbol S_j in finite precision due to fixed-point noise b_x as

$$P_{x_i,j} \quad = \quad \text{Prob}\{\hat{x}_{ip} + b_x \in R_j \big| \hat{x}_{ip} \in R_i\}.$$

The quantization noise is uncorrelated with the signal [120] and hence the two events can be expanded in terms of probability density function as

$$P_{x_i,j} \quad = \quad \lim_{\epsilon \to 0} \int_{x_i - \frac{\epsilon}{2}}^{x_i + \frac{\epsilon}{2}} f_x(x_i)dx_i \cdot \int_{R_j} f_b(b - x_i)db. \tag{4.15}$$

Considering all the points in region R_i, the total error probability $P_{i,j}$ is given by

$$\begin{aligned} P_{i,j} \quad &= \quad \int_{R_i} P_{x_i,j} dx_i. \\ P_{i,j} \quad &= \quad \int_{R_i} \left[f_x(x_i) \int_{R_j} f_b(b - x_i)db \right] dx_i. \end{aligned} \tag{4.16}$$

By re-arranging the terms in Equation 4.16, the error $P_{i,j}$ can be written as

$$P_{i,j} = \int_{R_i} \int_{R_j} f_x(x_i) f_b(b - x_i) db dx_i. \tag{4.17}$$

From Equation 4.17, the addition of quantization noise and the signal at the input of the decision operator can be identified region wise. $f_{T_i}(b)$ is the actual signal being presented at the input of the decision operator corresponding to all values taken by infinite precisoin signal in region R_i. For any region R_i, the PDF of the total quantization noise contribution is given as

$$f_{T_i(b)} = \int_{R_i} f_x(x_i) f_b(b - x_i) dx_i. \tag{4.18}$$

The total probability of error $P_{i,j}$ can be written in terms of $f_{T_i(b)}$ by substituting

the same in Equation 4.14 as

$$P_{i,j} = \int_{R_j} f_{Ti}(b)db. \tag{4.19}$$

The error matrix obtained by calculating $P_{i,j}$ is essentially the joint probability distribution of the decision outputs with and without fixed-point noise. That is

$$f_{\tilde{x}_{ip},\tilde{x}_{fp}}(\tilde{x}_{ip} = S_i, \tilde{x}_{fp} = S_j) = P_{i,j}. \tag{4.20}$$

The actual output signal probability distribution in the case of fixed-point noise can be obtained as marginal distributions of the joint distribution obtained thus far.

Probability Mass Function

The quantization error is best described by its probability distribution. Since the output of the decision operator is discrete, the error distribution is referred to as the probability mass function (PMF). The range of the PMF of the error at the output of the decision operator depends on the values assigned to the decision symbol set S. Further, it is assumed that on the decision set $S \in \{S_1, S_2, ..., S_L\}$ it is possible to define a distance measure $(d_{i,j})$ between any two values in S. It is also required that this distance operation is closed in the space where the symbols themselves are defined. For example, the distance measure can be simple euclidean distance between the two values as defined in Equation 4.21.

$$d_{i,j} = S_i - S_j. \tag{4.21}$$

The individual decision errors are represented by discrete time delta function $\delta(\tilde{x})$[1]. The strength of the delta function is made proportional to the probability of error. The decision error PMF $f_{\tilde{x}}(\tilde{x})$ is obtained as

$$f_{\tilde{x}}(\tilde{x}) = \sum_{j=1}^{L}\sum_{i=1}^{L} f_{\tilde{x}_{ip},\tilde{x}_{fp}}(\tilde{x}_{ip} = S_i, \tilde{x}_{fp} = S_j) \cdot \delta(\tilde{x} - d_{i,j}). \tag{4.22}$$

The function $\delta(\tilde{x} - d_{i,j})$ is a shifted version of the δ function in the two dimensional space. The range of this space consists of $L \times L$ discrete points as defined by various permutations of the output symbol set S consisting of L elements. Each $\tilde{x}_{i,j}$ corresponds to the error in decision generated for every pair (i, j) and the function $f_{\tilde{x}_{ip},\tilde{x}_{fp}}(\tilde{x}_{ip} = S_i, \tilde{x}_{fp} = S_j)$ is nothing but the error probabilities obtained as $P_{i,j}$ in Equation 4.17.

The correctness can be observed by comparing the PMF obtained by analytical expression and the result obtained by simulation. One way to theoretically verify the correctness of the expression in Equation 4.22 is to check if the sum of all individual probabilities is 1.

[1] Kronecker delta function

By substituting Equation 4.19 for $P_{i,j}$ and given that x and b are independent, the sum of all values in the PMF is given as

$$\sum_{i=1}^{L}\sum_{j=1}^{L} P_{i,j} = \sum_{i=1}^{L}\int_{R_i} f_X(x_i)\cdot \underbrace{\sum_{j=1}^{L}\int_{R_j} f_B(b-x_i)\cdot db}_{=1}\cdot dx_i. \tag{4.23}$$

By definition, the area under a PDF is always 1. The inner most summation in Equation 4.23 is only a shifted version of the PDF of the quantization noise variable. Therefore it evaluates to 1. Applying the same principle on the outer summation, the sum of probability $P_{i,j}$ over the range $L \times L$ evaluates to 1. In other words, the probability mass funtion of decision errors given in Equation 4.22 is a valid PMF.

Total Decision Error Probability

This study of decision errors with L regions throws up L^2 different combinations of probabilities to be calculated. Among these, the L probabilities where the decision operators give the same output in both fixed precision and infinite precision case are not errors. The rest are errors and contribute to the total error rate due to quantization.

The error at the output of the slicer is obtained by summing up all the $P_{ij}\ \forall\ (i,j)\ \in [1, L]$ and given by

$$P_{error} = \sum_{i=1}^{L}\sum_{j\neq i, j=1}^{L} P_{i,j}. \tag{4.24}$$

The final expression for total error probability is then obtained by plugging in Equation 4.16 into Equation 4.24 as

$$P_{error} = 1 - \sum_{j=1}^{L} P_{j,j}. \tag{4.25}$$

Complexity Analysis

To calculate the strength of the discrete Dirac function for every possible error $P_{i,j}$ at the output of the first decision operator requires the knowledge of signal statistics apart from the statistics of the accumulated quantization error. The signal statistics is acquired by carrying out one high precision simulation or alternatively by using some semi-analytical techniques based on such as [4] as discussed in Section 2.2. The quantization noise statistics can be derived analytically by using the single noise source (SNS) model described in Chapter 3.

If there are N symbols generated at the output of the decision or un-smooth quantization operator, $N(N-1)$ number of combinations need to be considered to obtain the total error. For each of the combinations, it is required to evaluate the double integral in Equation 4.17. Numerical evaluation of the double integral could be quite

complex and its computational complexity depends on the number of samples used to represent the actual PDF. Clearly, the complexity of calculating the error probability at the output of an un-smooth operator is of the order $O(N^2)$.

4.5.2 Propagating Decision Errors

Very often, it may be the case that there are more than one un-smooth quantizer in the system. The technique for deriving the error probabilities in the previous section is used to analytically determine un-smooth error statistics only in the absence of other un-smooth errors. To make this method practically useful, it is necessary to develop techniques for propagating such errors through other smooth and un-smooth operators is equally important.

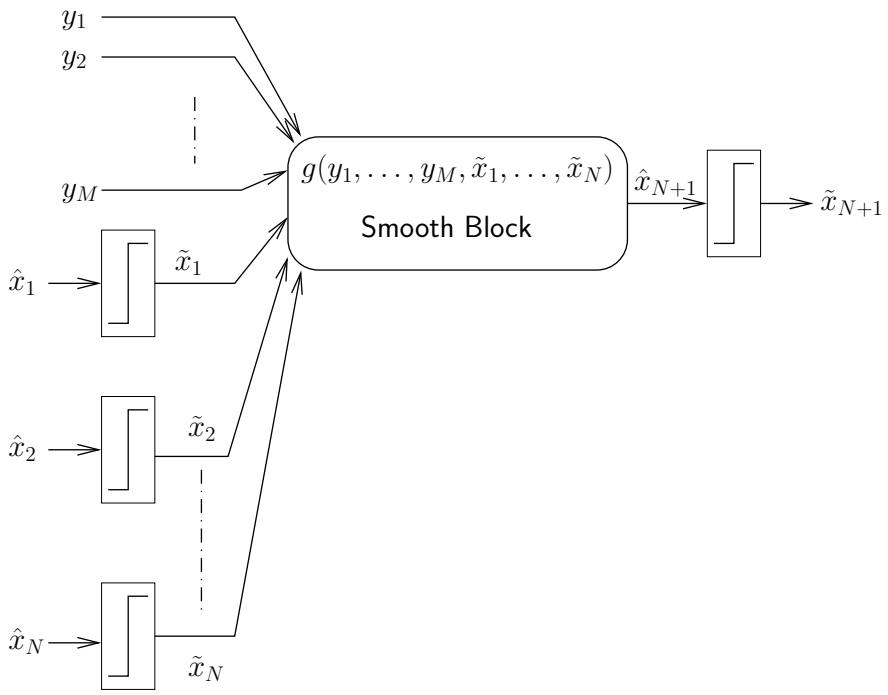


Figure 4.13: Propagating PMF across un-smooth operators

Consider the system level block diagram as shown in Figure 4.13. This block diagram consists of all possible scenarios in which un-smooth quantizer is used. There are $N+1$ un-smooth operators shown in the block diagram. The computational block evaluates a smooth function $g()$ with $N+M$ inputs of which, the N inputs are sourced from the outputs of un-smooth quantizers.

In a fixed-point implementation of the system, the input vector $\mathbf{y} = [y_1, \ldots, y_M]$ are perturbed by the quantization noise vector $\mathbf{b} = [b_1, \ldots, b_M]$ where each element in the vector corresponds to the noise perturbation of the input signal with the respective index. Let vectors $\mathbf{y}^{ip}$ and $\mathbf{y}^{fp}$ be the values taken by M inputs to the function $g()$

in infinite precision and finite precision implementations respectively. Likewise, the values assumed by the decision inputs $\tilde{\mathbf{x}} = [\tilde{x}_1, \ldots \tilde{x}_N]$ in a fixed-point system can be different when compared to the values taken in an infinite precision implementation. Let vectors $\tilde{\mathbf{x}}^{ip}$ and $\tilde{\mathbf{x}}^{fp}$ be the values assumed by these decision outputs in infinite precision and finite precision implementations respectively. Correspondingly, let $\hat{x}^{ip}_{N+1}$ and $\hat{x}^{fp}_{N+1}$ respectively be the infinite and finite precision values of the signal at the output of the smooth block $g()$. Then, the expression for the signal $\hat{x}^{fp}_{N+1}$ at the output of the smooth block is given as

$$\begin{aligned}
\hat{x}^{fp}_{N+1} &= g(y^{fp}_1, \ldots, y^{fp}_M, \tilde{x}^{fp}_1, \ldots, \tilde{x}^{fp}_N), \\
&= g(y^{ip}_1, \ldots, y^{ip}_M, \tilde{x}^{ip}_1, \ldots, \tilde{x}^{ip}_N) \\
&\quad + G(b_1, \ldots, b_M, \tilde{x}^{ip}_1, \tilde{x}^{fp}_1, \ldots, \tilde{x}^{ip}_N, \tilde{x}^{fp}_N) + b_g, \\
&= \hat{x}^{i}_{N+1} + g_b(\mathbf{b}, \tilde{\mathbf{x}}^{ip}, \tilde{\mathbf{x}}^{fp}) + b_g, \qquad (4.26)
\end{aligned}$$

where b_g is the noise added by the smooth quantizers used in the fixed-point implementation of the function block $g()$ and $g_b()$ is the corresponding noise propagation function.

Given that $g()$ is a smooth function, the noise propagation function $g_b()$ is linear with respect to the perturbation vector $\mathbf{b}$. The propagation of errors at the output of N un-smooth decision operators through the function $g()$ may not be characterized linearly without losing generality. Therefore, the noise propagation function $g_b()$ has to be evaluated for each and every type of error that can occur at the output of each of the N decision operators.

Consider an instance where $\tilde{\mathbf{x}}^{ip}$ and $\tilde{\mathbf{x}}^{fp}$ takes on particular values of $\tilde{\mathbf{x}}^{IP}$ and $\tilde{\mathbf{x}}^{FP}$ respectively. The signal $\tilde{\mathbf{x}}^{fp}_{N+1}$ is then given as

$$\begin{aligned}
\hat{x}^{fp}_{N+1} &= \hat{x}^{ip}_{N+1} + g^{IP,FP}_b(b) + b_g, \\
&= \hat{x}^{ip}_{N+1} + b_f, \qquad (4.27)
\end{aligned}$$

where the noise propagation function $g^{IP,FP}_b$ is derived from the function $g_b()$ in Equation 4.26 by using the particular values of $\tilde{\mathbf{x}}^{IP}$ and $\tilde{\mathbf{x}}^{FP}$ in the place of $\tilde{\mathbf{x}}^{ip}$ and $\tilde{\mathbf{x}}^{fp}$ respectively. Therefore, the function $g_b()$ reduces to a function of perturbation vector $\mathbf{b}$ only. It addresses the issue of propagating the perturbations associated with the input signals $\mathbf{y}$ through the function $g()$ for a given combination of input decision vector combination in infinite and finite precision implementations. The decision input vectors in finite and infinite precision jointly represent a scenario $\mathbb{C}$ of input conditions. The noise generated due to fixed-point operations in implementing the function $g()$ and the propagation of the input perturbations can be summed up and collected at the output of the function $g()$ as noise b_f.

If there can be L_k decision symbols that can be assigned to the output of the k^{th}

decision output, the vector $\tilde{\mathbf{x}}$ can assume as many as $C = \Pi_{k=1}^{N} L_k$ unique combinations. Let the vector $\mathbb{C}_N^\kappa$ be defined as $\mathbb{C}_N^\kappa = [\tilde{\mathbf{x}}^{IP}, \tilde{\mathbf{x}}^{FP}]$ where the elements of the vector take on different particular values. The number of unique combinations of the vector $\mathbb{C}_N^\kappa$ are as many as $C \times C$. The particular choice of $\tilde{\mathbf{x}}^{IP}$ and $\tilde{\mathbf{x}}^{FP}$ represents one combination $\mathbb{C}_N^\kappa$ which can completely define the input scenario of choice.

The probability of error at the output of the $(N+1)^{th}$ decision operator for a given combination of decision input vector combinations $\mathbb{C}_N^\kappa$ is calculated as $P_{ij}^{\mathbb{C}_N^\kappa}$ which can be calculated by using the technique presented in the previous section as

$$P_{i,j}^{\mathbb{C}_N^\kappa} = \int_{R_i} \int_{R_j} (f_x^{\mathbb{C}_N^\kappa}(x) f_b(b-x) db) dx, \tag{4.28}$$

where $P_{i,j}^{\mathbb{C}_N^\kappa}$ is the probability that the value of $\tilde{x}_{N+1}$ which takes the value S_j in infinite precision and the value of S_i in a finite precision implementation under the given input decision scenario defined by $\mathbb{C}_N^\kappa$. The function $f_x^{\mathbb{C}_N^\kappa}(x)^{ip}$ corresponds to the PDF of the signal $\hat{x}_{N+1}^{ip}$ for the scenario $\mathbb{C}_N^\kappa$ and the function $f_B(b)$ corresponds to the total quantization noise PDF b_f obtained in the Equation 4.27.

The total error of probability can be calculated by considering the effect of all possible scenarios on the output decision. That is, by taking a weighted summation of the probability of occurrence of all possible vectors $\mathbb{C}_N^\kappa$ and their impact on the output of the decision errors as

$$\begin{aligned} P_{i,j} &= \sum_{\kappa=1}^{C \times C} P_{i,j}^{\mathbb{C}_N^\kappa} \cdot P_{\mathbb{C}_N^\kappa}, \\ &= \sum_{\kappa=1}^{K} \int_{R_i} \int_{R_j} P_{\mathbb{C}_N^\kappa} (f_X^{\mathbb{C}_N^\kappa}(x) f_b(b-x) db dx), \end{aligned} \tag{4.29}$$

where $P_{\mathbb{C}_N^\kappa}$ is the probability of occurrence of the decision output scenarios as given by the vector $\mathbb{C}_N^\kappa$.

Calculating $P_{\mathbb{C}_N^\kappa}$: Independent Decision Errors

The probability of occurrence of the scenario $\mathbb{C}_N^\kappa$ is obtained as the joint probability that a combination of decision operator outputs occurs according to the description of the scenario. That is,

$$P_{\mathbb{C}_N^\kappa} = P((\tilde{x}_1^{fp} == S_i^1, \tilde{x}_1^{ip} == S_j^1) \wedge \ldots \wedge (\tilde{x}_N^{fp} == S_i^N, \tilde{x}_N^{ip} == S_j^N)), \tag{4.30}$$

where S_i^t and S_j^t represent the value taken by the t^{th} decision operator in infinite and finite precision implementations respectively. The $\wedge$ operation represents logical *and* operation. In other words, $P_{\mathbb{C}_N^\kappa}$ is the probability of occurrence of the κ^{th} scenario. If

the elements of the vector $\tilde{\mathbf{x}}$ are un-correlated with one another, the joint probability is obtained as

$$P_{\mathbb{C}_N^\kappa} = \Pi_{t=1}^{N} P(\tilde{x}_t^{ip} == S_i^t, \tilde{x}_t^{fp} == S_j^t). \tag{4.31}$$

That is, it is sufficient to know the error probabilities as discussed in Section 4.5.1 at the output of each decision operator.

Calculating $P_{\mathbb{C}_N^\kappa}$: Dependent Decision Errors

In practice, there can be instances where some of the decision outputs are correlated with one another. The correlation is determined by the signal flow topology and the correlation that exists between the input test vectors used to perform the high precision simulation for acquiring the signal PDF. Under such circumstances, the influence of each of the decision outputs on other decision outputs need to be considered in order to calculate the joint probability of occurrence of the decision outputs in the scenario $\mathbb{C}_N^\kappa$.

Let $\mathbb{C}_P^\kappa$ be defined as a sub-scenario: a subset of the scenario $\mathbb{C}_N^\kappa$. Then, the elements of the vectors $\tilde{\mathbf{x}}^{ip}$ and $\tilde{\mathbf{x}}^{fp}$ are of length P ($P \leq N$) and have the same value as the first P elements defined in the vector $\mathbb{C}_N^\kappa$. The joint probability $P_{\mathbb{C}_N^\kappa}$ can be evaluated by considering the conditional probabilities as

$$\begin{aligned}
P_{\mathbb{C}_N^\kappa} &= P((\tilde{x}_N^{fp} == S_i, \tilde{x}_N^{ip} == S_j) \mid \mathbb{C}_{N-1}^\kappa) \cdot P_{\mathbb{C}_{N-1}^\kappa}. \\
&= P((\tilde{x}_N^{fp} == S_i, \tilde{x}_N^{ip} == S_j) \mid \mathbb{C}_{N-1}^\kappa) \cdot \\
&\quad P((\tilde{x}_{N-1}^{fp} == S_i, \tilde{x}_{N-1}^{ip} == S_j) \mid \mathbb{C}_{N-2}^\kappa) \cdot \\
&\quad \vdots \\
&\quad \cdot P((\tilde{x}_2^{fp} == S_i, \tilde{x}_2^{ip} == S_j) \mid \mathbb{C}_1^\kappa) \cdot \underbrace{P((\tilde{x}_1^{fp} == S_i, \tilde{x}_1^{ip} == S_j))}_{P_{\mathbb{C}_1^\kappa}}. \\
&= \Pi_{i=0}^{N-1} P((\tilde{x}_{N-i}^{fp} == S_i, \tilde{x}_{N-i}^{ip} == S_j) \mid \mathbb{C}_{N-i-1}^\kappa).
\end{aligned} \tag{4.32}$$

From Equations 4.31 and 4.32, it is clear that the calculation of the probability of the scenario $P_{\mathbb{C}_N^\kappa}$ requires the knowledge of the error probabilities and joint probabilities of the N decision inputs. This requires that the probability of error values $P(\tilde{x}^{ip} == S_i, \tilde{x}^{fp} == S_j)$ of the decision outputs be computed apriori. This essentially means that, the probability of error at the output of every decision operator in the system has to be calculated in precedence order starting from the input to the output of the system. To do this, such dependencies must be traced by parsing the system graph soon after identification of the un-smooth operators.

When the decision inputs are not correlated with one another, the joint probability is obtained as a simple product of the various individual decision scenarios in infinite and finite precisions. On the other hand when decision operator outputs are correlated with one another, the calculation of the error PMF for the first slicer is obtained by

technique described in Equation 4.22 in the previous section. The subsequent quantizer PMF is calculated by calculating the joint probability of errors successively as described in Equation 4.32. In the process, the intermediate results obtained during calculation of PMF at the output of a given un-smooth quantizer is re-used for calculation of scenario probabilities in successive quantizers.

Calculating $P_{\mathbb{C}_N^\kappa}$: Generic Case

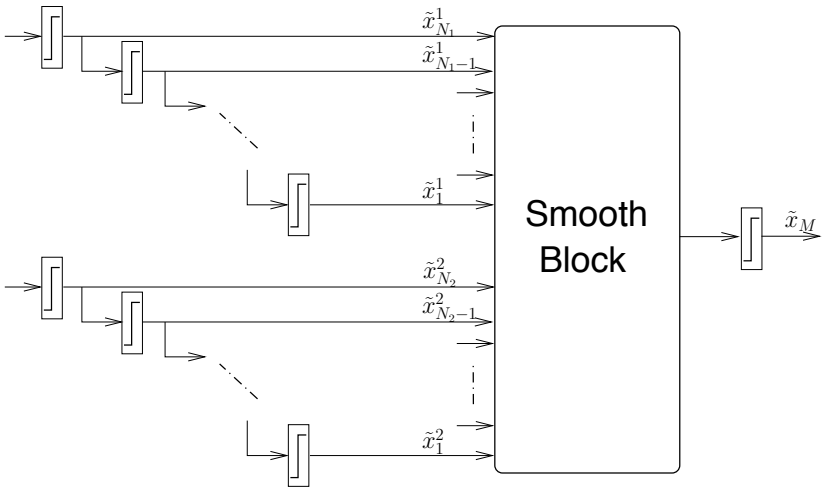


Figure 4.14: Propagating correlated and un-correlated un-smooth errors

For the sake of illustrating the generic nature of the proposed error PMF propagation technique, consider a smooth sub-system block which has two groups of un-smooth inputs where the signal in one of the groups is independent of signals from the other but is correlated with a signal in the same group as shown in Figure 4.14. Let the signal of the decision operator at the otuput of the smooth block be $\tilde{x}_M$. That is, the N inputs are segregated into two independent groups consisting of N_1 and N_2 decision operators. Any i^{th} un-smooth operator is dependent on $(i-1)^{th}$ operator in the respective clusters. That is $(\tilde{x}^1_{N_1} \rightarrow \tilde{x}^1_{N_1-1} \rightarrow \ldots \rightarrow \tilde{x}^1_1)$ and $(\tilde{x}^2_{N_2} \rightarrow \tilde{x}^2_{N_2-1} \rightarrow \ldots \rightarrow \tilde{x}^2_1)$, where $\rightarrow$ indicates the dependency between two un-smooth quantizers.

The objective is to be able to analytically calculate the probability of occurrence of all possible scenarios resulting from $N_1 + N_2$ number of inputs. The various input scenarios are defined by all possible un-smooth operator inputs. That is the scenario $P_{\mathbb{C}^\kappa_{N_1+N_2}}$ considers the different combinations of values from N_1+N_2 un-smooth inputs. Since the two groups of signals are uncorrelated with one another, the probability of occurrence of these scenarios can be obtained by the product of scenarios of the groups separately and scenarios $P_{\mathbb{C}^\kappa_{N_1}}$ and $P_{\mathbb{C}^\kappa_{N_2}}$ are expanded as given by Equation 4.32 individually. Therefore, the total probability of input scenario considering all the different un-smooth operator inputs can be calculated as

$$
\begin{aligned}
P_{\mathbb{C}^{\kappa}_{N_1+N_2}} &= P_{\mathbb{C}^{\kappa}_{N_1}} P_{\mathbb{C}^{\kappa}_{N_2}}. \\
&= \left\{ \Pi_{i=0}^{N_1-1} P((\tilde{x}^{fp}_{N_1-i} == S_i) \&\& (\tilde{x}^{ip}_{N_1-i} == S_j) \mid \mathbb{C}^{\kappa}_{N_1-i-1}) \right\} \cdot \\
&\quad \left\{ \Pi_{i=0}^{N_2-1} P((\tilde{x}^{fp}_{N_2-i} == S_i) \&\& (\tilde{x}^{ip}_{N_2-i} == S_j) \mid \mathbb{C}^{\kappa}_{N_2-i-1}) \right\}. \quad (4.33)
\end{aligned}
$$

Complexity Analysis

The calculation of one conditional probability as shown in Equation 4.28 is similar to the derivation of the probability $P_{i,j}$ of the first decision operator as discussed in the previous section. The effort to calculate the joint probability can be measured in units of evaluating the integral numerically which calculates the values of $P_{i,j}$ for a given scenario. The number of times this has to be calculated depends on the number of unique scenarios.

The number of unique scenarios grows exponentially with the number of un-smooth decision inputs that need to be taken into consideration. In both correlated and un-correlated un-smooth cases, evaluation of these many integrals is inevitable given the number of combinations these errors would generate. Suppose if there are L_k number of symbols at the output of the k^{th} un-smooth quantizer, it contributes to L_k unique scenario combinations. Considering all the combinations of un-smooth operator outputs, there can be $C = \Pi_{k=1}^{N} L_k$ of them in total. Therefore, the complexity of calculating the error probabilities at the output of the $(N+1)^{th}$ quantizer which depends on the previous N un-smooth quantizers is of the order of $O(C^2)$, where C has an exponential complexity.

4.5.3 Computing Probability of Error in Communication Systems

Estimating BER in a matched receiver

Consider a baseband communication system with a matched filter receiver as shown in Figure 4.15. The quantization noise in the system is shown as an additive source in the signal flow graph.

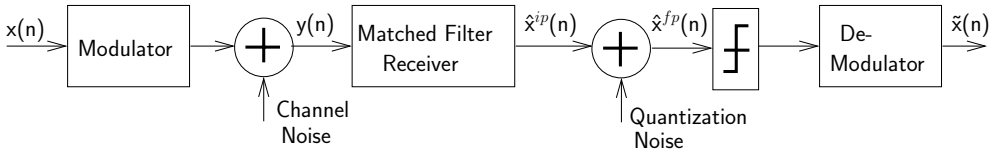


Figure 4.15: Base-band communication: matched filter receiver system

In Figure 4.16, the error PMF obtained by taking the Euclidean difference between the symbols obtained at the output of the decision operator in infinite precision and finite precision is shown. The 16-QAM scheme is chosen for transmission of data in this experiment. The channel noise power has unit variance and the quantization noise

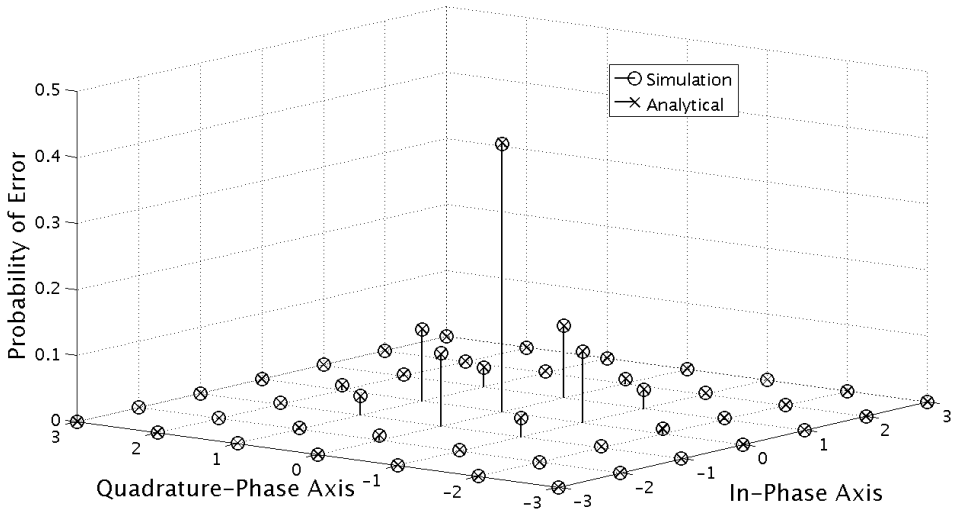


Figure 4.16: Probability Mass Function of signl $\tilde{x}(n)$ at the output of the decision operator

power is equal to 0.5. The error PMF at the output of the slicer which is obtained by simulation closely matches with the PMF obtained analytically. The maximum error between probability mass function obtained by simulation and by analytical formula is not more than 3%.

Estimating Error PMF for MIMO Sphere Decoder (V-Blast)

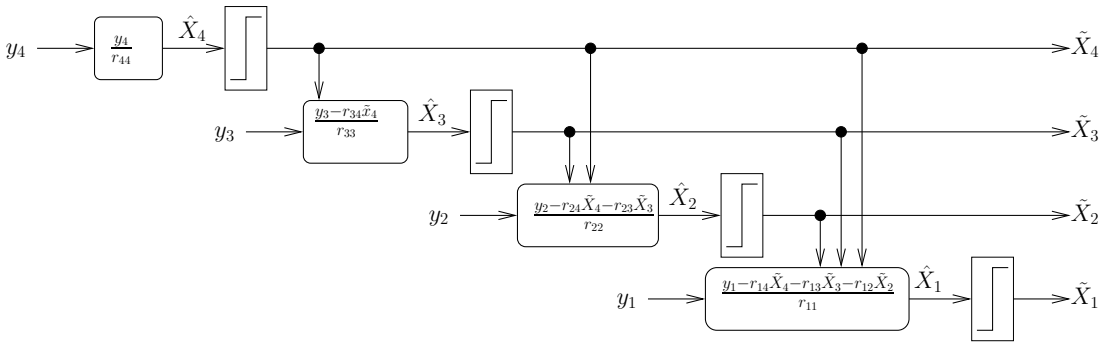


Figure 4.17: V-Blast: data-flow model and associated smooth blocks

The signal flow graph of the V-BLAST algorithm used for MIMO decoding as shown in Figure 4.17 is considered. The V-BLAST algorithm with four Receiver antennas has four un-smooth quantizers or decision operators. The symbols participating in the

system are drawn from an unbiased BPSK source. It is assumed that these symbols are transmitted through an additive white Gaussian noisy channel. The inputs y_i in case of computational block corresponding to every antenna is therefore a Gaussian noise. For the sake of simplicity, it is assumed that the channel matrix is an upper triangular R matrix. Therefore, all inputs y_i for all values of i have the same standard deviation.

In general, this is not true as the channel matrix is not upper-triangular. Practically, the channel matrix $\mathbf{H}$ is QR-decomposed as $\mathbf{H} = \mathbf{QR}$, where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is the upper triangular matrix. The input signals received are multiplied by the transpose of the orthogonal matrix before processing it with the V-BLAST computation SFG shown in Figure 4.17. This changes the noise power of the input y_i in V-BLAST. However, this experiment does not make any assumption about the equality of channel noise and hence, this is representative of the processing carried out in practice.

The total cumulative noise-perturbation is considered to have a Gaussian distribution with zero mean in case of all four antennas. In order to keep the experimentation simple in terms of numerical evaluation, the actual Kurtosis values are ignored and a standard Gaussian is used for the quantization noise. Therefore, The Gaussian input signal PDF and quantization noise PDF are given as

$$f_{y_i}(y) = \frac{1}{\sqrt{2\pi}\sigma_{ch}} \left\{ e^{-\frac{(y)^2}{2\sigma_{ch}^2}} \right\} f_B(b_x) = \frac{1}{\sqrt{2\pi}\sigma_q} \left\{ e^{-\frac{(b_x)^2}{2\sigma_q^2}} \right\}, \tag{4.34}$$

where σ_{ch} and σ_q are the standard deviation of the noisy channel and quantization noise respectively.

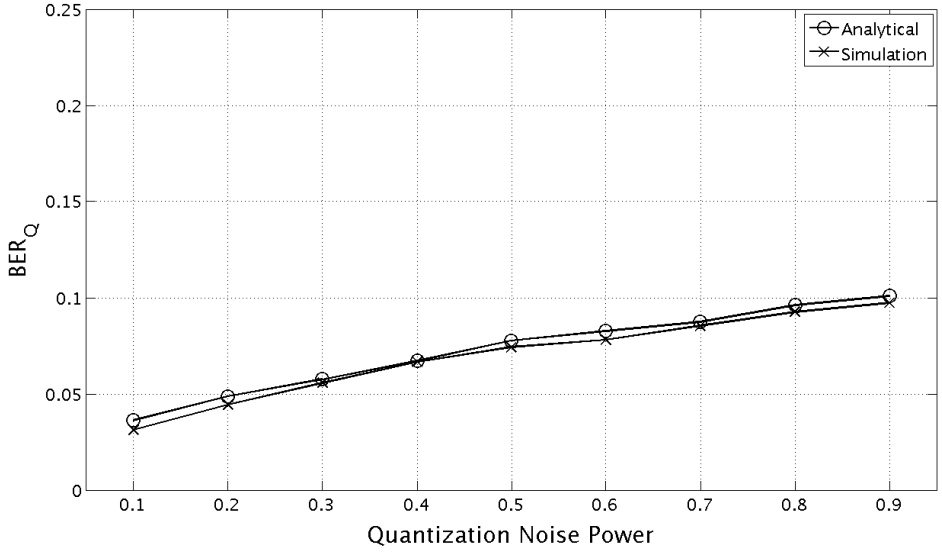


Figure 4.18: Decision Error rate of V-BLAST at antenna 4

To obtain the results, the standard deviation of channel noise is set to 1 and the

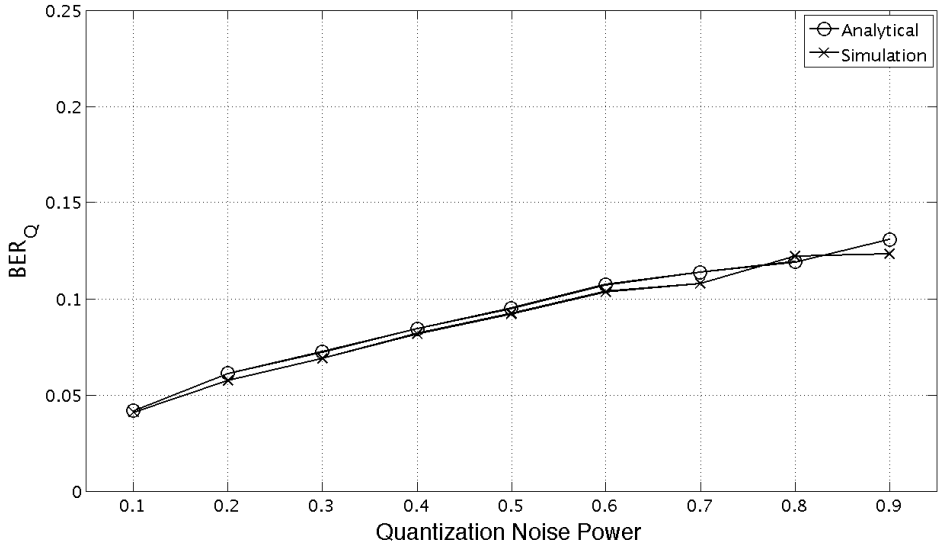


Figure 4.19: Decision Error rate of V-BLAST at antenna 3

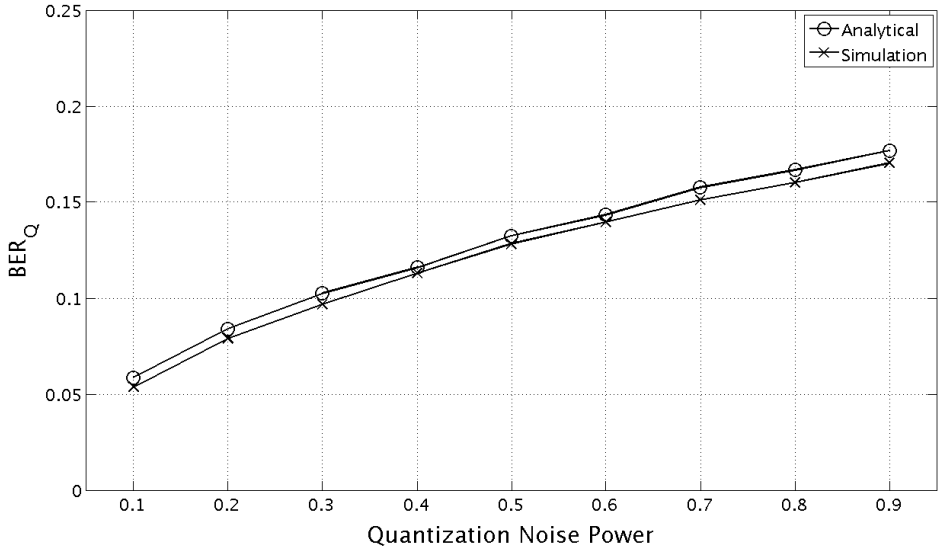


Figure 4.20: Decision Error rate of V-BLAST at antenna 2

quantization noise power is varied from 0.1 in steps of 0.1 upto 0.8. The corresponding increase in bit error rate contributed only due to quantization noise is plotted along the vertical axis as BER_{Q}. The results obtained by using the analytical technique and by simulation for the decision error at the output of each of these antennas are given in Figures 4.18, 4.19, 4.20 and 4.21.

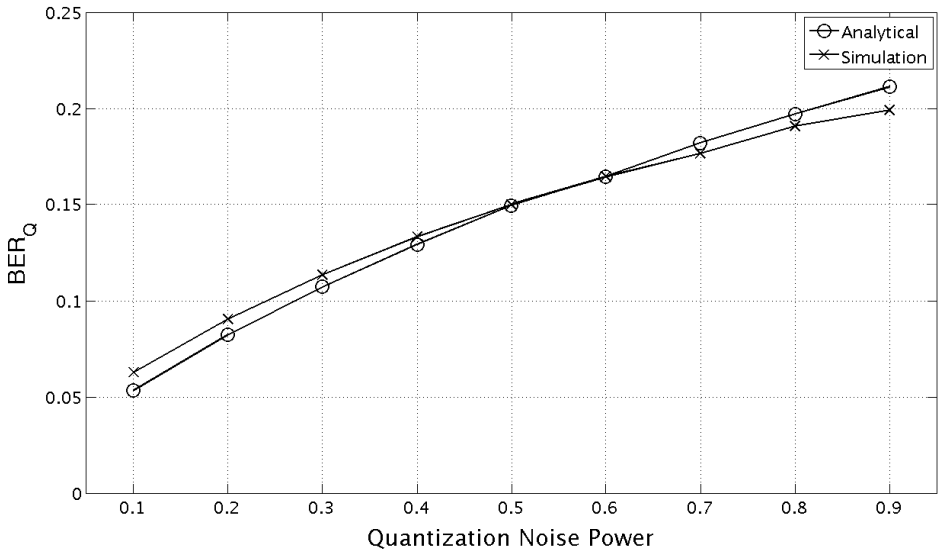


Figure 4.21: Decision Error rate of V-BLAST at antenna 1

The PMF of the decision operator output corresponding to the 4^{th} antenna is obtained by evaluating the expression in Equation 4.22 for various error values. In order to generate all possible scenarios at the input of computational block corresponding to antenna 3, it is required to consider the $(P^4_{i,j})$ values of antenna 4. The probability $(P^4_{i,j}, P^3_{i,j})$ of scenarios at the input of smooth block corresponding to antenna 2 is obtained by applying Equation 4.32. The probability of different scenarios at the input of computational block corresponding to antenna 1, that is the joint probability of $(P^4_{i,j}, P^3_{i,j}, P^2_{i,j})$ is obtained similarly. The total PMF of the signal at the output is calculated by plugging in the probability numbers into Equation 4.29.

The results obtained analytically closely match the results obtained by simulation. The deviation from simulaiton results changes differ between the different antennas. In most cases, this is not beyond 5% in magnitude. The accuracy of results obtained by analytical technique depends on the resolution with which the signal and noise PDF is discretized. Also, the results obtained by simulation heavily depends on the number of sample points used. The deviation between the results obtained by analytical and simulation are essentially due to this experimental artefact. The quality of the result increases with increasing resolution of the discretization of PDFs and the number of points used for simulation. The increase in BER_{Q} in successive antennas is due to the addition of quantization noise as a result of using fixed-point arithmetic. The accuracy of results obtained depend heavily on the numerical integration technique for calculating Equations 4.22 and 4.28. In this experiment, the integration was carried out numerically by discretizing the signal and quantization noise PDF and calculating region-wise convolution effectively evaluating $f_{T_i}(b)$ for every region for all combinations.

4.6 The Hybrid Technique

Thus far, the idea of un-smoothness has been discussed in the realm of quantizers with large step-sizes alone. However, there are other operators whose response to perturbation by quantization noise cannot be always captured by analytical expressions. These operators can be commonly found in situations where data statistics are used for influencing the control-flow. The *Min()* and *Max()* operators are good examples for such kind of un-smooth operators. These operators are used to determine the minimum and maximum values respectively from a pool of data inputs. There are no external control inputs in this case to influence the behavior of the operator. Even then, statistical prediction of the outcome of minimum value and maximum value operator in spite of the knowledge of signal statistics is difficult when there are more than two inputs [103].

Apart from the challenges posed by un-smooth operators whose fixed-point behavior does not have an analytical model, it is also possible that the response to fixed-point inputs for certain types of sub-systems cannot be modeled analytically. Such situations can arise when designing mixed-signal systems where some of the sub-systems are implemented using analog circuits and they interact with digital inputs and outputs. Under such circumstances the sub-system functionality has to be given a black-box treatment and therefore it becomes inevitable to use fixed-point simulation for evaluating the performance of fixed-point sub-systems.

During fixed-point simulation, all operations are simulated using a fixed-point library. If the system under consideration consists of smooth operators only, it is possible to avoid performing fixed-point simulation by using the analytical techniques discussed in Chapter 3. However, the presence of even one un-smooth operator or sub-system whose fixed-point behavior cannot be captured makes the use of fixed-point simulation for the entire system inevitable. In such situations, a few un-smooth operators become obstacles for taking the benefit of analytical models.

Therefore, the focus here is on using the analytical SNS model selectively along with floating-point simulation to reduce the total fixed-point simulation time. In this section, sub-systems or operators whose response to fixed-point behavior is not derived analytically are referred to as un-smooth blocks or un-smooth operators respectively.

The SNS model is selectively applied only to those sub-systems which consist of smooth operators only. The perturbations generated by the SNS model is statistically equivalent to the fixed-point error obtained by simulation. The same effect is recreated by adding the perturbations generated by the SNS model to the nominal signal recorded during simulation in infinite precision. Thereby, using the SNS model instead of fixed-point computation greatly reduces the computational effort even while keeping the experiment statistically consistent. This reduction in the time/ resource spent on fixed-point computation is realised as acceleration of fixed-point simulation.

The idea of accelerating fixed-point simulation requires careful consideration before application on practical systems. These considerations are first discussed in the rest of this section before presenting the details of the algorithm 4.3 in Section 4.7.

4.6.1 Preparation for Hybrid Evaluation

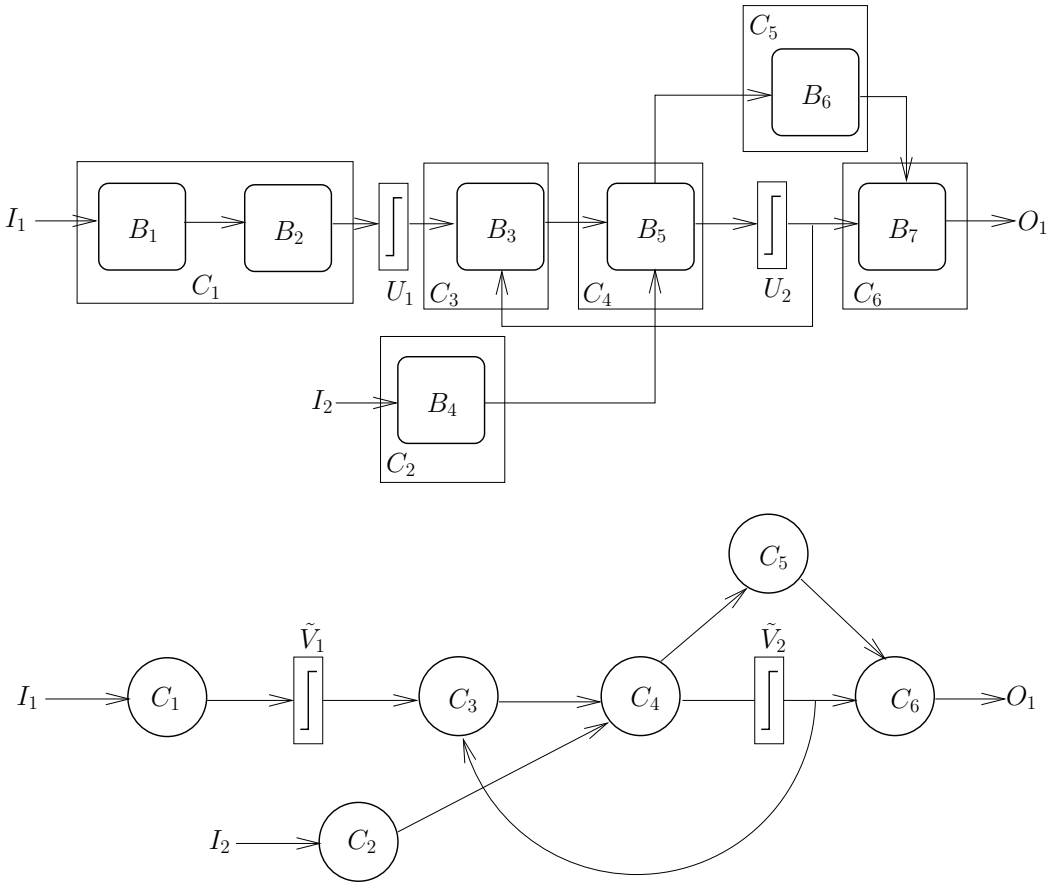


Figure 4.22: A representative signal processing system

Consider a representative signal processing system with two un-smooth operators and seven smooth blocks as shown in Figure 4.22. It is possible to derive the single noise source model for analysis of quantization noise contribution at the output of each smooth block. With the help of the *transmit filter* of the SNS model, these values can be propagated through other smooth blocks as long as no un-smooth operator is encountered. The first step in applying the selective evaluation is to partition the given graph into smooth clusters separated by un-smooth operators. In the example considered in Figure 4.22, there are no un-smooth operators in between blocks B_1 and B_2. Therefore, they can be clustered together. An erroneous un-smooth input to a smooth blocks affects the output of other smooth blocks. For example, blocks B_3 and B_5 cannot be combined together as B_3 has an input from both un-smooth operators U_2 and U_1 and B_5 does not have any un-smooth input. Likewise, B_4, B_5 and B_6 cannot be combined together as it might be required to perform simulation of B_5 and also B_6 due to an un-smooth error whereas B_4 does not require simulation. The cluster graph

obtained by combining smooth blocks is also shown in Figure 4.22.

The contribution in this part of the thesis is not about the most efficient way of identifying these clusters. Rather, it is about applying the hybrid simulation technique on a given signal flow graph where arithmetic clusters are clearly identified and separated from un-smooth operators. The idea behind using the proposed hybrid technique is to make maximum use of the available SNS model for propagating and estimating the fixed-point noise behavior of all smooth blocks B_1 to B_7 instead of performing fixed-point simulation of the entire system.

4.6.2 Acceleration by Selective Evaluation

The hybrid technique is proposed to accelerate the process of fixed-point simulation. When the hybrid technique is applied instead of fixed-point simulation, not all fixed-point operations are simulated. Instead the decision to perform fixed-point simulation or to use the analytical method in each iteration is subject to the signal conditions prevailing during that particular simulation. The decision to simulate is taken only if there is a likelihood of occurrence un-smooth errors. This reduction in simulation effort effectively leads to acceleration of the fixed-point simulation process.

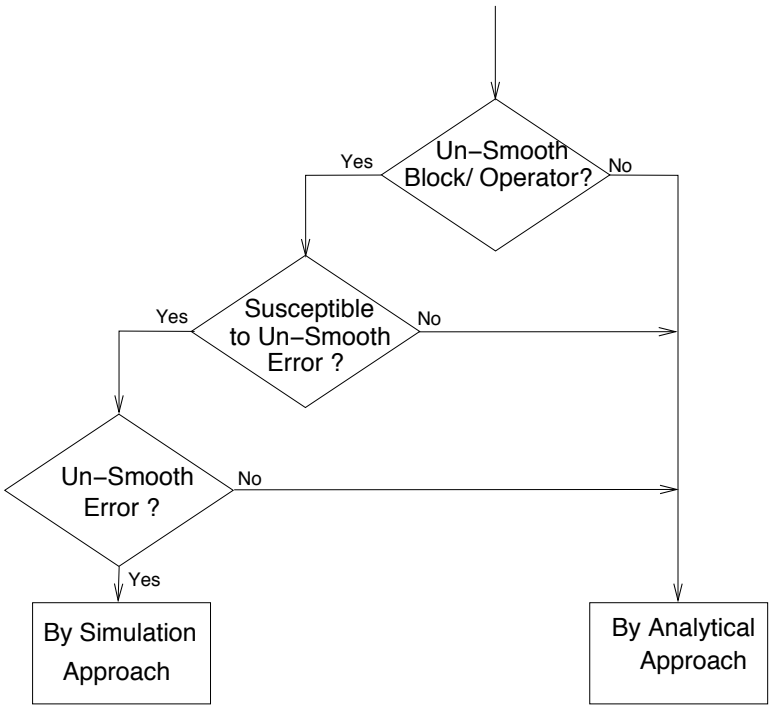


Figure 4.23: Decision flow diagram for conditional evaluation

The decision flow diagram shown in Figure 4.23 describes the decision making process. This decision tree is executed at the input of every un-smooth operator once corresponding to every iteration of fixed-point simulation. In a given iteration, the

cluster graph under consideration is traversed. During this traversal, if an un-smooth operator or an un-smooth block is encountered, it is checked if an un-smooth error can occur in the presence of quantization noise obtained by the Single Noise Source model. This check takes into account the signal conditions prevailing in the particular iteration of interest. If the output is not susceptible for an error, the nominal values obtained at the output of the un-smooth operator are valid and there is no need to perform simulation.

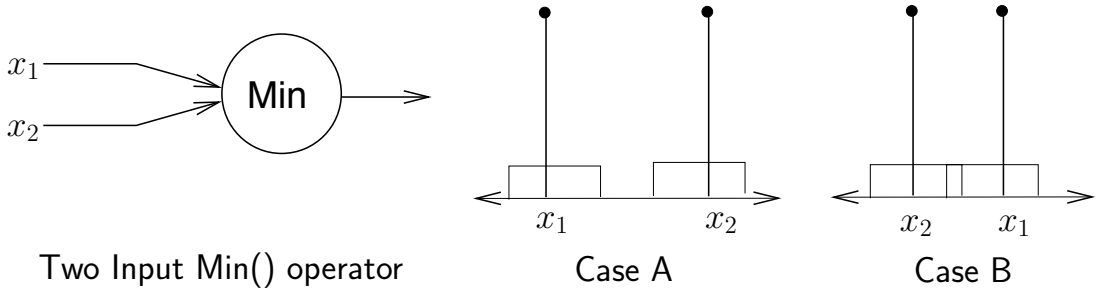


Figure 4.24: Min(): an un-smooth operator

In some cases, it is possible to deduce that un-smooth errors cannot occur owing to the nominal value and the quantization noise power obtained by the SNS models at the input of the given un-smooth operator. One such scenario in the case of un-smooth quantizers is described in Figure 4.11(case B). Another example with respect to the *Min()* operator which takes two inputs is shown in Figure 4.24.

Here again, two scenarios are considered. In *case A*, nominally x_1 is smaller than x_2 and the two values are separated enough for the quantization noise not to affect output. In *case B*, nominally, x_2 is smaller than x_1 but they are closely spaced for the given quantization noise power. In this scenario it is hard to say without simulation which of the two inputs is chosen as minimum. If x_1 is chosen instead of x_2, it leads to an error which cannot be determined without actually knowing the values taken by the signals.

If the susceptibility to un-smooth error is indeterminable, a random number is generated using the random process defined by the SNS model and it is checked if an un-smooth error occurs by simulating the un-smooth operator only. If there are no un-smooth errors, the nominal value at the un-smooth output is valid and the noise can be propagated analytically. Otherwise, the rest of the system has to be simulated at least for that particular iteration in which the un-smooth error has occurred.

4.6.3 Lifetime of an Un-smooth Error

In a system with no memory, the un-smooth error does not affect successive iterations. Therefore, it is not required to consider the occurrences of un-smooth errors in previous iterations during simulation. In such simple cases, the application of selective fixed-point simulation is straight forward. The nominal data acquired during the high

precision simulation serves as a reference value of the signal during every simulation iteration. The output of the smooth clusters in finite precision can be obtained by perturbing the nominal output.

However, the presence of memory in practical systems requires that the effect of an un-smooth error is considered not only in the simulation iteration in which the error occurred but in successive iterations where the effect of un-smooth errors continue to influence the computations. The number of iterations an un-smooth error affects the output of a smooth cluster is referred to as the *Lifetime* of the un-smooth error of the given un-smooth block. The *Lifetime* of a cluster can be determined by observing the topology of the interconnect between various operations within the cluster. The *Lifetime* of a cluster between any of the inputs and outputs is defined as the number of memory elements in the path between them. If a cluster has more than one input and output, each combination can have a particular value of *Lifetime*.

Considering the topology of the interconnect between clusters and the decision operators, two unique scenarios: one without any feed-back and the other with feed-back are considered. These scenarios are representative of all possible practical scenarios of interconnect between sub-systems. These two scenarios are represented by two topologies in Figure 4.25. In both cases, there are two smooth clusters (C_1 and C_2) and two un-smooth operators U_1 and U_2. The topology of the individual clusters (C_1 and C_2) could be either feed-forward (FF) or feed-back (FB) in nature. In either case, the first cluster C_1 does not require simulation. The second cluster needs to be simulated depending upon the values of the signal at the output of un-smooth operator U_1.

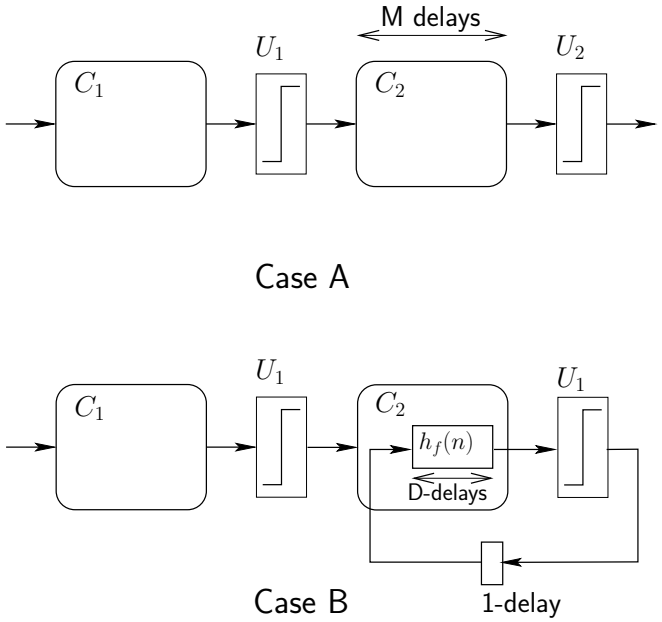


Figure 4.25: Two topologies: Case A: feed-forward and Case B: feed-back

If the second cluster C_2 has a feed-forward topology, the effect of decision error at

the input is flushed out after a deterministic number of samples. This can be determined by counting the number of delays on the path from the input to the output of the second cluster C_2. If the cluster C_2 has a feed-back structure such as an IIR[1] filter, the error remains in the sub-system indefinitely. In such cases, the *pseudo-impulse response*[2] of the path function between the input and the output of cluster C_2 provides the number of iterations after which the effect of the un-smooth error can be ignored. The *Lifetime* M, of the cluster C_2 in *case A* between clusters U_1 and U_2 is determined by evaluating the pseudo-impulse response of the path path function if it consists of feed-back loops or by the number of memory elements on the longest delay path between the input and the output if the topology of cluster C_2 is feed-forward. In other words, the cluster C_2 needs to be simulated for M subsequent samples starting from the sample in which the error was generated in un-smooth operator U_1. If another un-smooth error occurs at the output of U_1 before M samples have passed, the simulation continues until M more samples have passed from that instant. If the cluster C_2 does not contain any memory, M takes a value of 0. Then, the cluster is simulated only for that sample where the error occurs.

The evaluation of *Lifetime* of clusters for *case B* is different from *case A* as a loop which feeds back the decision output of the un-smooth operator U_2 back into cluster C_2 exists. Let the feed-back path function $h_f(n)$ consist of D delay elements witin the cluster and one delay element outside of it. An error at the output of the operator U_2 affects the output of the cluster C_2 from the next sample. This is because the error is held in the delay outside the cluster during the present cycle. Hence, the simulation of cluster C_2 following an error at the output of un-smooth error U_2 in the next iteration. In a signal processing system, a feed-back loop must contain a delay element to avoid race conditions (sometimes also referred to as negative edge cycles [104]). Therefore, the value of the *Lifetime* for the feed-back path is at least 1 when there is a decision output is in the feed back path. For *case B* considered in Figure 4.25, the *Lifetime* of the corresponding cluster is $D + 1$. The extra 1 unit for the *Lifetime* in this case is contributed by the delay in the feed-back path outside the cluster. That is, the system in *case B* requires $D + 1$ successive samples to be simulated starting from the sample when an error occurs at the output of U_2.

4.6.4 The Simulation Subgraph

Having determined the *Lifetime* of un-smooth errors across each input-output pair of all systems, the next task is to identify the sub-graph of the given cluster graph which requires fixed-point simulation in the event of an un-smooth error. The extent to which an un-smooth error affects the rest of the system depends upon its locality. The subgraph consisting of nodes and edges that get affected due to an error at the output of a particular un-smooth operator can be deduced by analysing the cluster graph. A procedure for obtaining the subgraph of the node starting from the un-smooth operator

[1] Infinite Impulse Response

[2] pseudo-impulse-response is an equivalent representation of the given filter with an FIR filter

node where the decision error has occurred is given in Algorithm 4.2.

Procedure 4.2 :GetSubgraph (t, H)

```
\* Classify Edges into: forward edges (E_f) and backward edges (E_b) *\
[E_f E_b] = GetDfsEdges(H);
\* Get the forward subgraph for node t *\
T_t = GetSubBFStree(H, E_f, t);
V = {};  \* Start with an empty list L *\
for all V_i in T_t do
   L = L + V_i;  \* All nodes in the forward path are marked *\
end for  \* Nodes in the loop need to be marked too *\
E_t = GetOutEdges(t);
for all e ∈ E_t do
   if e ∈ E_b then
      ν = e.GetDestinationNode();  \* Get the back edge node in ν *\
      L = L+ GetSubgraph(ν);  \* Mark subgraph of ν for simulation *\
   end if
end for
Return L;
```

The signal flow graph is a directed graph and therefore it is possible to identify the *forward* edges ($\mathbb{E}_f$) and *backward* edges ($\mathbb{E}_b$) using the BFS[1] algorithm. Once the edges are identified, the BFS tree is defined by the *forward* edges in a connected signal flow graph. In case of signal flow graph which does not contain loops, the subgraph can be easily identified by considering the connected nodes between the given node t in the BFS tree formed by the forward edges in the BFS graph.

The back edges are considered for identifying the subgraph of the nodes present in a signal flow graphs with feed-back loops. If one of the edges emanating from the node under consideration is also a back edge of the signal flow graph, the subgraph of the destination node is also marked for simulation.

In the signal flow graph example considered in Figure 4.22, all the edges except the edge joining nodes $\tilde{V}_2$ and C_3 are BFS-forward edges. Consider two different cases: in the first case, an error occurs at the output of the un-smooth operator $\tilde{V}_1$ and in the second an error occurs at the output of the un-smooth operator $\tilde{V}_2$. The subgraphs of nodes $\tilde{V}_1$ and $\tilde{V}_2$ are shown in Figure 4.26.

In the first case (subgraph of $\tilde{V}_1$), there are no back edges emanating from node $\tilde{V}_1$. Therefore, the nodes in the subgraph are marked for simulation. In the second case, the edge joining nodes $\tilde{V}_2$ and C_6 is a forward edge. The back edge $\tilde{V}_2 \rightarrow C_3$ is considered and the other nodes are marked for simulation following the subgraph of node C_3.

The nodes C_3, C_4 and C_5 would already be simulated by the time node $\tilde{V}_2$ is simulated and hence they are marked for simulation in the next iteration. Whereas, the simulation of node C_6 begins with the present iteration itself. This is also the expected

[1]Breadth First Search

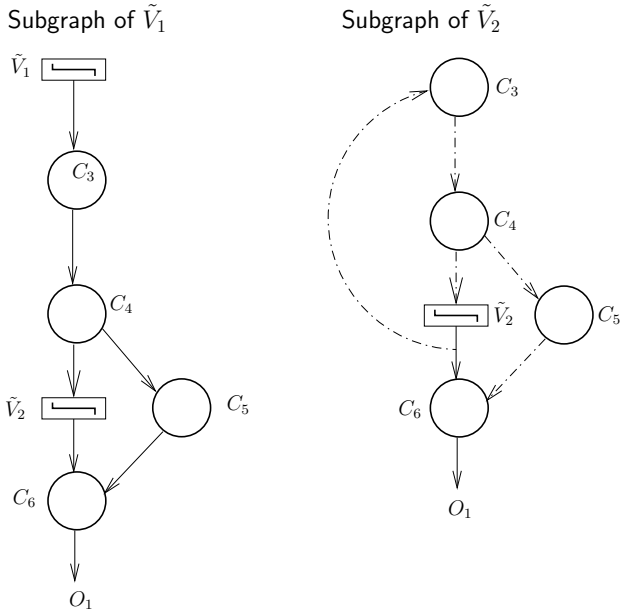


Figure 4.26: subgraph of nodes $\tilde{V}_1$ and $\tilde{V}_2$

behavior as an error in the given iteration at the output of node $\tilde{V}_2$ is separated by at least one delay element in the feed-back loop and hence does not affect the computations in the present iteration.

4.7 Hybrid Simulation Algorithm

The hybrid simulation algorithm is given in Procedure 4.3. It essentially consists of two main phases. The first among them is the *Pre-processing* phase. This is a one time effort and is unique for a given signal flow graph and a given set of input data. The second phase is where the actual hybrid simulation is performed. This phase is as good as the fixed-point simulation process in its functionality and therefore, this phase is repeated for every given word-length assignment.

4.7.1 Pre-processing Phase

The pre-processing step is a one-time effort for a given signal flow graph of the algorithm, a given set of input test vectors and for a given minimum number of bits assigned to each of the operations. In the case of a given signal flow graph, certain operations such as $Min()$ or $Max()$ that do not have an analytical model to predict the response to perturbation are by definition un-smooth. In addition, Algorithm 4.1 *Identifying Un-smooth Boundaries*, is used for determining the bounds on the word-length formats where the PQN model can be applied. With the knowledge of minimum number of bits that can be assigned to each of the operations in the system, the operators can be

Algorithm 4.3 :**Accelerated Evaluation**

* **Pre-Processing:** *\
Identify un-smooth Boundaries; * *Execute Algorithm 4.1* *\
for all nodes V_i in $G(V, E)$ **do**
* *Identify un-smooth operators: Large step-size or by definition* *\
if $(minBits \leq V_i$.unsmoothBitBoundary) || (V_i is un-smooth) **then**
V_i.unsmooth = $true$;
end if
end for
$\mathbf{H}(\mathbf{C}, \tilde{\mathbf{V}}, \tilde{\mathbf{E}})$ = GetClusterGraph($G(V, E)$);
for all Cluster $C_i \in H$ **do**
C_i.DeriveSNSModel(); * *SNS model for Analytical Noise Propagation* *\
C_i.EstimateLifetimeValue(); * *Latency to purge an un-smooth error* *\
C_i.SimMode = $false$; * *In Analytical mode by default* *\
C_i.ResetLifetimeCounter(); * *No un-smooth errors* *\
end for
for all un-smooth operators $\tilde{V}_i \in H$ **do**
$S(V, E)$ = GetSubTree($\tilde{V}_i, H$) * *Get the sub-graph of each un-smooth operator* *\
$\tilde{V}_i.S = S(V, E)$; * *Store the sub-graph for use during simulation* *\
end for
* **Hybrid Simulation:** *\
for all $n \in N$: Input test-vector **do**
while Entire $\mathbf{H}(\mathbf{C}, \tilde{\mathbf{V}}, \tilde{\mathbf{E}})$ is not simulated **do**
T = GetReadyNodes($H(C, \tilde{V}, \tilde{E})$); * *Nodes whose tokens are available* *\
for all $t \in T$ **do**
* *Check if the node t is an un-smooth operator* *\
if ($t_i \in \tilde{V}$) && (t_i.SimMode == $false$) **then**
SimMode = EvaluateDecisionTree(t, n, H);
if SimMode == $true$ **then**
SetSimMode(t); * *Set subgraph of node t to simulation mode* *\
end if
else
* *Check if the cluster is in simulation mode* *\
if t_i.SimMode == $true$ **then**
SimErrorProp(t, H);
else
SNSErrorProp(t);
end if
end if
end for
end while
Mark $\mathbf{H}(\mathbf{C}, \tilde{\mathbf{V}}, \tilde{\mathbf{E}})$ as un-simulated;
end for

classified as smooth or un-smooth. This process generally requires one high-precision simulation of the signal flow graph with the given input test vectors. The acquired high-precision data corresponding to signals at the boundary of un-smooth operators are stored for future reference and the rest of the acquired data is discarded. The hybrid technique can then be used without having to repeat this pre-processing steps for any number of repeated evaluations with different fixed-point formats as long as the fixed-point quantizers continue to be smooth. In case of the example in Figure 4.22, there are two un-smooth operators and the data corresponding to the input and the output of these un-smooth operators need to be stored.

All the smooth operators are collected together to form smooth clusters. The original graph $\mathbf{G}(\mathbf{V}, \mathbf{E})$ with V nodes, where each node corresponds to a smooth sub-system and is interconnected by E edges is transformed to an equivalent *Cluster Graph* $\mathbf{H}(\mathbf{C}, \tilde{\mathbf{V}}, \tilde{\mathbf{E}})$. The cluster graph consists of C smooth clusters, $\tilde{V}$ un-smooth operators or un-smooth blocks and $\tilde{E}$ edges inter connecting the clusters and the un-smooth operators. In the system considered in Figure 4.22, the blocks B_1, B_2 are clustered to form cluster C_1 while the other blocks are individual clusters C_2 to C_6. The single noise source (SNS) model corresponding to each of the clusters C_i in the graph is derived. This activity can be completely parallelized. The subgraph corresponding to each of the un-smooth operator is identified by executing Algorithm 4.2. Also, a double-precision simulation is carried out and the nominal values of the signals at the input and output of un-smooth blocks are stored in a signal database.

4.7.2 Hybrid Simulation Phase

To effectively apply the proposed selective simulation technique, it is assumed that the default system behavior is a small perturbation from the values obtained during infinite precision and there shall be no errors at the output of un-smooth operators. Therefore, counter to keep track of the *lifetime* of the error in every cluster is reset to zero and the associated flag suggesting simulation mode is set to *false*. The idea here is to treat the occurrence of an error as a pathological case and that the analytical models are good enough otherwise.

The cluster graph $H(C, \tilde{V}, \tilde{E})$ represents the signal flow graph. The simulation of a signal flow graph is often considered as a system processing tokens. Where, each input test case is treated as a token. Therefore, one token is consumed and one token is generated at the output of every sub-system in every iteration. The set of N test vectors is considered one by one for simulation of the algorithm. In other words, the test vector set consists of N tokens and one token is considered in every iteration and is processed through the signal flow graph.

The input token activates a set of nodes T in the cluster graph H. These nodes consume the input token and produce one token at their outputs. The new set of ready nodes T are the ones for which the available tokens are sufficient for processing. This process is repeated while the entire signal flow graph $H(C, \tilde{V}, \tilde{E})$ is not covered. When the entire signal-flow graph is covered once, it marks the end of passing one token through the system. The same procedure is repeated for the next token and repeated

until all tokens are processed.

Checking for Un-smooth Errors

To decide whether an error can occur at the output of the un-smooth operator, the decision tree in Figure 4.23 is traversed by considering the reference signal at the input of every un-smooth operator.

The algorithm for evaluating the decision tree is provided in Procedure 4.3a. Here, the first step is to first identify the clusters from which the un-smooth operator is drawing its inputs. This is obtained by simply looking at the *in-edges* of the node representing the un-smooth operator t. Then, the reference samples corresponding to the iteration under consideration is obtained from the data-base. Let C_i be the cluster whose output is fed into the un-smooth operator t. Because of quantization noise, the nominal reference value of a signal $x_i(n)$ while processing the n^{th} token corresponding to the output of the cluster C_i would be perturbed by quantization noise b_{x_i} due to smooth fixed-point quantization. If there is a possibility of an un-smooth error, then a random value is generated from a random process corresponding to the SNS model of the cluster C_i and assigned to b_{x_i}. If this perturbation is large enough to cause an error at the output of the un-smooth operator, then *SimMode* is set to *true*. It is set to *false* otherwise.

Procedure 4.3a :EvaluateDecisionTree(t, n, H)

```
1:  \* Identify cluster at the input of un-smooth operator t *\
2:  C_i = GetFeedinCluster(t, H);
3:  x_i(n) = GetReferenceSample(n);   \* Obtain n^th reference sample *\
4:  if SusceptibleToUnsmoothError(x_i, b_{x_i}) then
5:     b_{x_i} = GenerateRandomQNoise(C_i);
6:     x̀_i = x_i + b_{x_i};
7:     if t(x̀_i) ≠ t(x_i) then
8:        SimMode = true;
9:     else
10:       SimMode = false;
11:    end if
12: else
13:    SimMode = false;
14: end if
15: Return SimMode;
```

Therefore, in scenarios such as *case A* in Figure 4.24, it is possible to set the flag *SimMode* to $false$ right away. Whereas, in *case B*, the random values have to be simulated to check whether an un-smooth error occurs. Even in this case, the flag *SimMode* is set to *true* only if an error actually occurs and not otherwise.

4.7.3 Propagating Simulation Mode

The correctness of the proposed hybrid algorithm heavily depends on performing simulation of the sub-systems whenever un-smooth errors are present. Only those subsystems that get affected by the un-smooth error shall be simulated. Such sub-systems are already identified during the pre-processing phase. The occurrence of an un-smooth error means that an input which is not the same as in the case of high precision is being presented at the input of the subsequent node. This potentially also means that the output of the subsequent clusters can also change and hence the clusters in entire subgraph is potentially presented with a different input.

Procedure 4.3b :SetSimMode(t)

1: $S(V,E) = t.S$; * *Obtain the sub-graph for node t* *\
2: **for all** $V_i \in S(V,E)$ **do**
3: V_i.SimMode = *true*;
4: V_i.SetLifetimeCounter(); * *Sets the latency count to* M_i *\
5: **end for**

Such sub-systems lie on the path starting from the node at which the un-smooth error occurred to the output of the system need to be simulated. When an un-smooth error occurs, the *SimMode* flag of all nodes (clusters) that belong to the subgraph of that un-smooth operator are marked *true* as shown in Procedure 4.3b. It means that the noise has to be propagated through all those nodes by simulation. Along with setting the flag, the *Lifetime Counter*, which essentially reflects the extent to which the error continues to affect the computations of the given cluster, is set to the pre-computed value as obtained in the pre-processing phase.

4.7.4 Evaluating Quantization Noise

The idea of conditional evaluation of sub-systems is central to the proposed hybrid simulation algorithm. If there are no un-smooth errors, the quantization noise behavior of the given system can be evaluated by the application of the SNS model to the reference values obtained by the high precision simulation data. In the hybrid approach, simulation is performed on sub-system only when there is a deviation from the reference values caused by the occurrence of un-smooth errors.

Procedure 4.3c :SNSErrorProp(t, n)

1: b_i = GetInputQnoiseParams(t);
2: b_o = EvalOutputQNoise(t, b_i); * *Evaluation by analytical method* *\

The quantization noise statistics obtained by the application of the SNS model is sufficient to statistically represent the quantization noise behavior when there are no un-smooth errors. The function $GetInputQnoiseParams()$ obtains the quantization noise

accumulated at the input of the sub-system and then evaluates the quantization noise at the output by using the function $EvalOutputQNoise()$ as shown in Procedure 4.3c.

Procedure 4.3d :SimErrorProp(t, H)

```
1: b_i = GetInputQnoiseParams(t);
2: b_o = Simulate(t, b_i);   \* Evaluation by simulation *\
3: t.DecrementLifetimeCounter();
4: SetSimMode(t, H);  \* Subgraph is simulated as long as errors are present *\
5: if t.GetCounterValue() == 0 then
6:    t.SimMode = false;
7: end if
```

When an un-smooth error occurs at the input of any cluster, it would be marked with a *SimMode* flag in Algorithm 4.3 to indicate the need for simulation. During simulation, it is also necessary to keep an account of the propagation of the injected un-smooth error. Therefore, in Procedure 4.3d, the *Lifetime Counter* associated with the cluster is decremented. As long as the cluster is simulated, an erroneous value (i.e. which is not the same as the reference value) continues to persist in the node t. Finally, the output of node t can potentially affect the computations of the nodes belonging to the subgraph of node t. Therefore, the flag *SimMode* is broadcasted to all the subgraph nodes as long as simulation continues.

When the *Lifetime Counter* becomes equal to zero, the erroneous value is purged out and the un-smooth operator outputs are similar to the value obtained in the reference simulation. Therefore, the flag *SimMode* of node t is reset. This change in the simulation mode flag should not be broadcasted to the subgraph of the node as there can be clusters in which the errors generated previously may continue to persist.

4.8 Performance of the Hybrid Technique

The time taken for performance evaluation is of interest when comparing the proposed hybrid technique with fixed-point simulation.

4.8.1 Fixed-point Simulation Effort

The computational effort involved in carrying out a fixed-point simulation is essentially that of performing all the computations in the system under consideration for each of the input test vectors using operators with assigned fixed-precision. In a system, if there are m different types of operators $O = \{O_1, O_2, \ldots O_m\}$ and if the test suite consists of N_t number of test vectors, the total time (T_{sim}) spent on simulation for estimation of fixed-point performance is given by

$$\begin{aligned} T_{sim} &= N_t \cdot \sum_{i=1}^{m} N_i \cdot \tau_i, \\ &= N_t \cdot t_{sim}, \end{aligned} \tag{4.35}$$

where τ_i is the time taken for simulating an operation of type O_i, N_i is the number of such operators in the system and t_{sim} is the time taken for simualation of all operations corresponding to one of the inputs in the test vector. It has to be noted that in a practical scenario, some of the operator types in O could be un-smooth or could be rendered un-smooth due to the assigned precision and the signal characteristics.

4.8.2 Hybrid Simulation Technique Effort

In the proposed hybrid approach, a finite amount of time is spent for the pre-processing overhead which includes identification of un-smooth operators, graph transformations and derivation of the SNS model for smooth clusters. Once the hybrid evaluation begins, parts of the system is simulated depending upon the occurrence of un-smooth errors. The total time for executing all the test cases can therefore be written as

$$T_{hyb} = t_{sns} + (N_a \cdot t_{ana} + N_s \cdot \tilde{t}_{sim}), \tag{4.36}$$

where t_{sns} is the time for performing the single noise source analysis. Some N_a of the total number of test vectors N_t are those cases in which there are no un-smooth errors and thus do not require any fixed-point simulation. The rest N_s are those that require parts of the system to be simulated depending upon the point of occurrence of un-smooth error. While t_{ana} is the time required to generate random numbers, the time for simulation in each individual case could vary. Here, $\tilde{t}_{sim}$ is the average simulation time required for simulating all those cases which require partial or complete simulation in fixed-point over N_s samples.

The value assumed by t_{ana} is very small in comparison to actual simulation as it involves generation of random numbers which can also be pre-computed. Also, the random numbers are used only in case the double precision reference signal happens to be in the error boundary according to the single noise source model and the generated signal. The value of $\tilde{t}_{sim}$ takes on a value anywhere between 0 and t_{sim} and is influenced by the number of decision errors when evaluating the input test vectors. Therefore, $\tilde{t}_{sim} \leq t_{sim}$.

With the application of the proposed hybrid technique, some of the samples N_s are simulated while the others N_a are handled analytically. In other words, the total number of samples is split between simulation and analytical modelling based evaluation ($N_t = N_s + N_a$). The time spent on pre-processing t_{sns} is a one-time effort performed once for a given signal flow graph. Thus in successive optimization efforts, this factor does not have any contribution to the time taken. Therefore, the total time spent on

simulation by using the proposed *Hybrid* approach can be approximated as

$$T_{hyb} \approx N_a \cdot t_{ana} + N_s \cdot \tilde{t}_{sim}. \tag{4.37}$$

The benefit obtained by following the hybrid approach can be quantified by an improvement factor (IF) which is defined as

$$\begin{aligned} IF &= N_a \cdot t_{ana} + N_s \cdot \tilde{t}_{sim}. \\ &= \frac{N_t \cdot t_{sim}}{N_a \cdot t_{ana} + N_s \cdot \tilde{t}_{sim}}. \end{aligned} \tag{4.38}$$

Typically, decision errors due to quantization noise is not very common and therefore $N_s << N_a$. Otherwise, the correctness of the system functionality using fixed-point operations is not acceptable (i.e. the performance of such system implemenations is not acceptable when there are too many errors due to fixed-point operations and such systems are not designed in practice). As a result, it can be seen that the improvement factor can take potentially large values. The maximum value of the improvement factor (IF) is limited by the generation of random numbers and checking for un-smooth errors. It is given by

$$IF_{max} = \frac{t_{sim}}{t_{ana}}. \tag{4.39}$$

The value IF_{max} is influenced by the number of un-smooth operators in the system. Large systems with relatively small number of un-smooth operators tend to have high improvement factors. As the precision of the fixed-point numbers increase, the improvement facto tends to the value of IF_{max}. On the other hand, the improvement factor IF decreases in value as the number of bits in precision is reduced until it assumes a value of 1. At that point, an un-smooth error occurs in the case of every input test vector.

4.8.3 Equivalence with Fixed-point Simulation

In the *Hybrid* approach, the noise at the output of a sub-system can be evaluated by simulation (when an error occurs on one of its un-smooth inputs) or the analytical models when there are no un-smooth errors. When an un-smooth error occurs during fixed-point simulation, it is because of quantization noise. As discussed in Section 4.5, the errors at the output of the un-smooth operator can be estimated with the knowledge of the signal and noise PDF. Therefore, it is possible to obtain the error probability using the *Hybrid* simulation approach by using the SNS model which is statistically equivalent to the quantization error. Thus, the errors obtained at the output of any of the un-smooth operators by employing the *Hybrid* approach are also statistically equivalent to un-smooth errors obtained by performing fixed-point simulation.

The proposed *Hybrid* simulation approach is after all only an alternative, but a fast alternative to performing fixed-point simulation of the system. Therefore, the statistical moments: mean, variance and higher order moments of the un-smooth outputs

can be obtained by a weighted average of the ratio of number of points evaluated by simulation and by analytical evaluation. In this thesis, the equivalence between the proposed *Hybrid* simulation approach and the fixed-point simulation errors is obtained by comparing the final metric at the system output used to measure accuracy of the system.

4.9 Application of Hybrid Technique

To show the effectiveness of the hybrid approach, the results obtained by applying the proposed technique on several synthetic and practical examples are presented. To begin with, a synthetic example is considered to illustrate the feed-forward topology shown in Figure 4.25. To illustrate the case where decision output is fed-back, the classical decision feed-back equalizer is considered. Then, the edge detection algorithm using the morphological operators [74] is studied. This example has the *Min()* and *Max()* operators as a part of morphological transformations. The third case is the SSFE algorithm [76]. This algorithm consists of many QAM slicers that are used for successive interference cancellation technique to equalize the MIMO baseband signals.

The proposed hybrid simulation technique is essentially as a faster alternative for fixed-point simulation. Therefore, it is as important to justify the statistical equivalence of this technique with fixed-point simulation as it is to show the improvement factors (IF) obtained. The results presented in each of the examples include plots to show the statistical equivalence between the proposed hybrid technique with fixed-point simulation and the improvement factors obtained thereof. The statistical equivalence is subject between the two approaches is subject to small errors due to the limited number of points considered for simulation and errors in estimation between the analytical and actual quantization errors.

4.9.1 Feed-forward Example

Consider the application of the hybrid technique on the feed-forward topology (case A) shown in Figure 4.25. The smooth clusters C_1 and C_2 are essentially linear filters. The quantization noise generation characteristics of each of these filters is obtained by the application of the Single Noise Source (SNS) model. The errors obtained at the output of each of the un-smooth operators by simulation and by employing the hybrid technique for various input bit-widths is shown in Figure 4.27. The input word-length is quantized is plotted on the x-axis and the symbol error rate at the output of each un-smooth operator is plotted on the y-axis. It has to be noted that the quantization noise contribution for precision value greater than 7 bits does not cause any error at the output of the second cluster and therefore not represented on the logarithmic scale. This figure essentially shows the equivalence between the hybrid technique and fixed-point simulation.

The improvement in time measured by the improvement factor as a result of using the hybrid technique for various quantization over fixed-point simulation is shown in

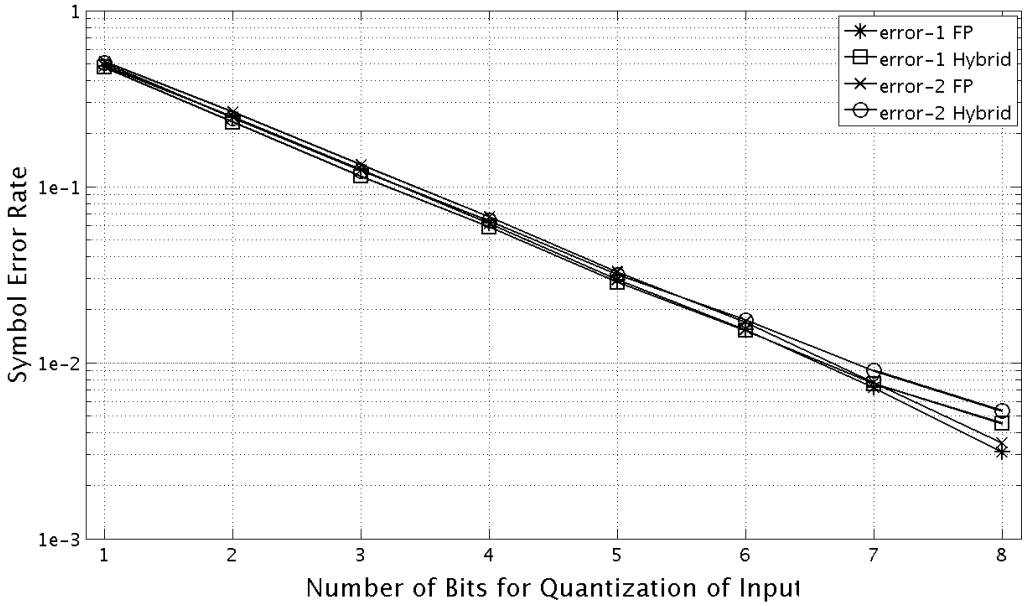


Figure 4.27: Feed-forward case: un-smooth error rates

Figure 4.28. It can be seen that the improvement factor indicates several orders of magnitude speed up in the time required for carrying out simulation. The speed up factor grows in magnitude as the quantization noise reduces.

4.9.2 Decision Feed-back Equalization

Decision feed-back equalizer (DFE) is a popular adaptive equalization technique used in various scenarios requiring error recovery. In this thesis, a basic is considered. The block diagram of this DFE is shown in Figure 4.29 is considered. It essentially consists of two arithmetic blocks, one feed-forward and the second feed-back marked as clusters C_1 and C_2. Both these blocks are essentially tapped-delay lines whose weights are adapted according to the LMS (least-mean-square equalization) algorithm by taking into consideration the values stored in the registers of the delay line in both feed-forward and feed-back blocks. The decision error is fed-back into the DFE through the feed-back path.

Figure 4.30 shows the equivalence between fixed-point simulation and the hybrid approach which makes use of the Single Noise Source model. The quantization noise generated from within the clusters is added using the SNS model at the adder that appears before the decision operator. Two sets of experiments are conducted where the fixed-point operations are uniformly assigned the same word-lengths. In the first case, the precision bits were set to 4 bits and it was changed to 6 bits in the second case. The relative error between the results obtained by fixed-point simulation and the

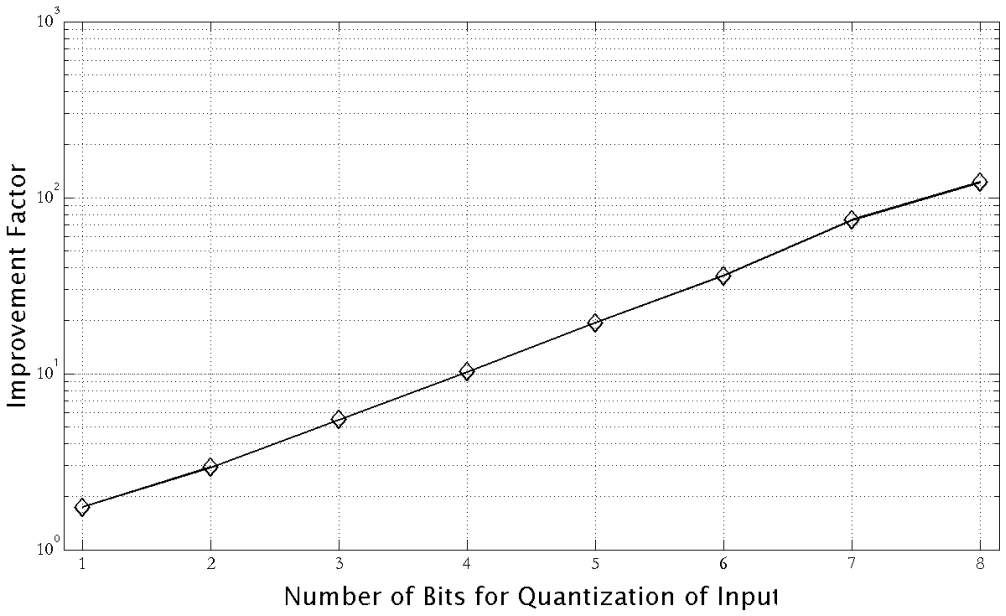


Figure 4.28: Feed-forward case: evolution of improvement factor (IF)

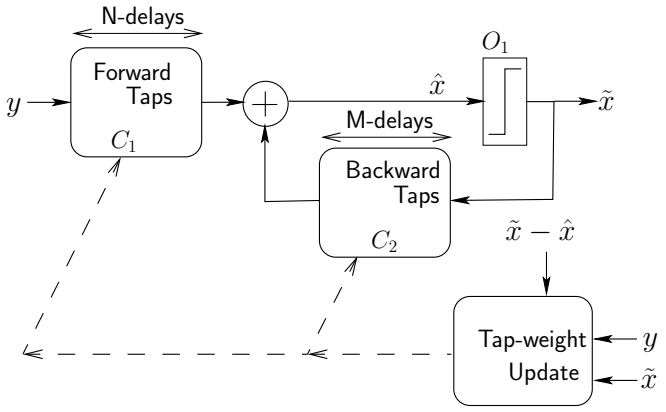


Figure 4.29: Decision Feedback Equalizer: Block Diagram

Hybrid simulation approach is as small as 2.5% and 4.2% in cases when 6 and 4 bits precision assigned to the fixed-point operations.

In Figure 4.31, the improvement factors obtained under various channel SNR conditions is shown. To better illustrate the trend, another experiment with 5-bit precision assigned uniformly across all fixed-point operations is considered. When the number of bits is less, the quantization noise is high and can hence cause more un-smooth errors. With increasing word-lengths, the number of un-smooth errors decreases. Therefore, as the word-lengths of fixed-point operations increase the number of instances which

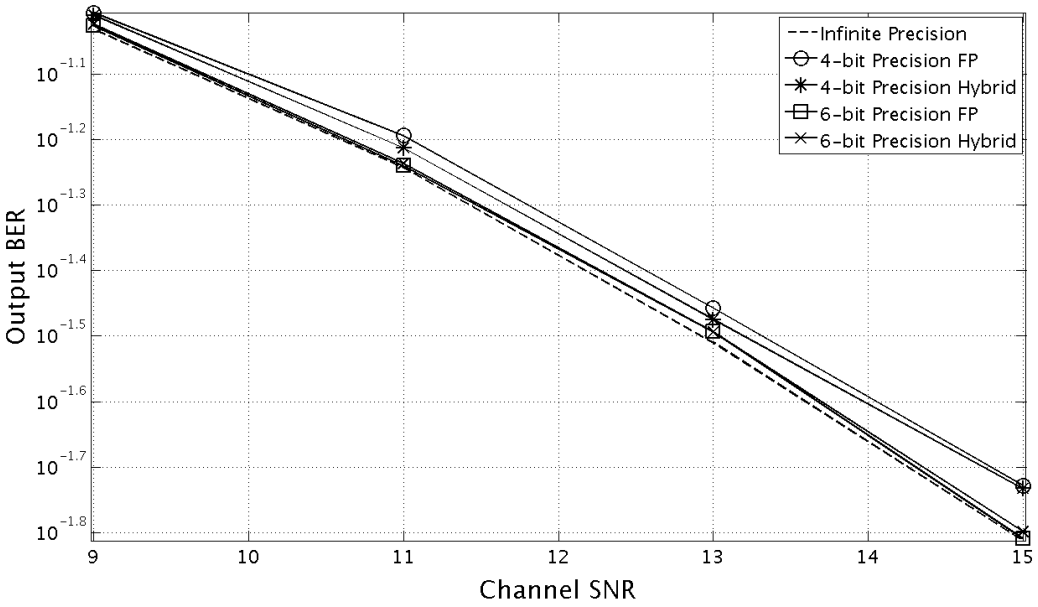


Figure 4.30: DFE: hybrid simulation equivalence with fixed-point simulation

require simulation decreases thereby contributing to a increase in the improvement factor (IF). This trend is also seen across varying channel noise. That is, the improvement factor increases with decreasing noise in the channel. The IF values achieved indicate speedup as high as two orders of magnitude.

4.9.3 Edge Detection

Detection of edges in a given image is a problem very frequently encountered in many image processing applications. Identifying edges in the image helps in object identification, motion tracking and image enhancement amongst many other applications of profound consequences. It is also often performed on blurred images in order to sharpen or restore image quality [113]. There are many edge detection schemes used in practice. For this experiment, one of the popular schemes which uses the *SOBEL* edge detector [105] followed by *thresholding* and morphological *Erosion* are applied successively on the input image in order to identify the edges. The schematic of the edge detection algorithm is shown in Figure 4.32.

The *SOBEL* edge detector is essentially a 2-dimensional linear high-pass filter. The quantization noise due to fixed-point effects can be captured using the SNS model. This is followed by image thresholding which detects the level of intensity of each pixel and transforms the image into a binary image according to the user defined threshold level. This transformation due to the threshold operator *Thresh* is given as

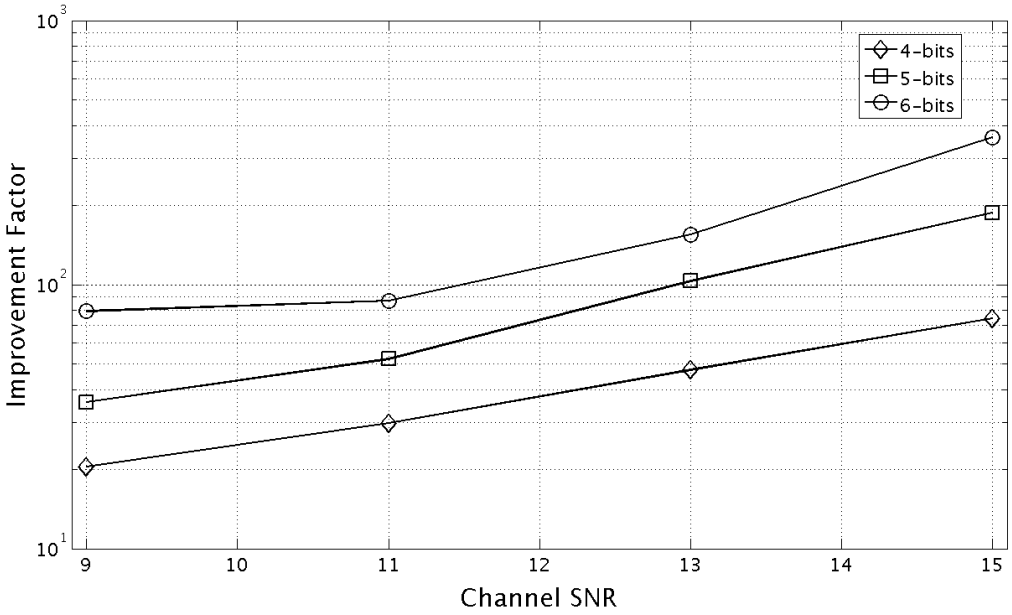


Figure 4.31: DFE: evolution of improvement factor (IF)

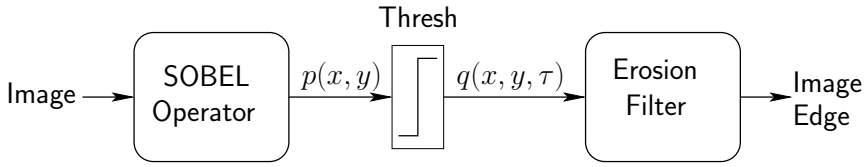


Figure 4.32: Edge Detection Schema

$$q(x,y,\tau) = \begin{cases} 0, & p(x,y) < \tau \\ 1, & p(x,y) \geq 0 \end{cases} \tag{4.40}$$

where $p(x,y)$ is the intensity value of the pixel at the position (x,y)[1] and τ is the threshold level which can either be user specified or calculated as the mean of the filtered image signal. The morphological *Erosion* operator is nothing but the $Min()$ operator applied using a user defined kernel throughout the image. In this experiment, the *diamond* kernel is used for carrying out the experiment. The bi-dimensionality of this kernel makes the edge detection sensitive to edges aligned in both vertical and horizontal detections.

Consider the application of the proposed *Hybrid* technique on this edge detection scheme. Of the three different operations considered, the *thresholding* operator and *thinning* sub-system are un-smooth. The *SOBEL* filtering is smooth and hence captured

[1]Here, x is the horizontal coordinate and y is the vertical coordinate

by the SNS model. The morphological transformations are particularly known to take a long time due to the sorting operation involved in determining the value of every pixel.

A double-precision simulation of the test images is carried out and the values at the output of the filter and the threshold operator is stored. Using the proposed hybrid simulation, if an error at the output of the threshold occurs, the morphological transformation corresponding to only that error has to be carried out. Hence, saving precious processing time in case of threshold values coinciding with that of the double precision simulation.

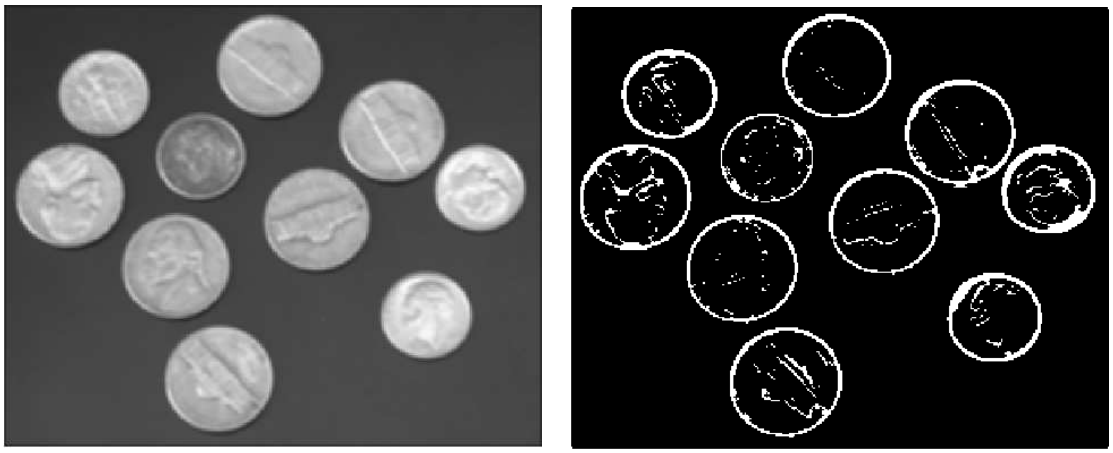


Figure 4.33: Coins image and its edges

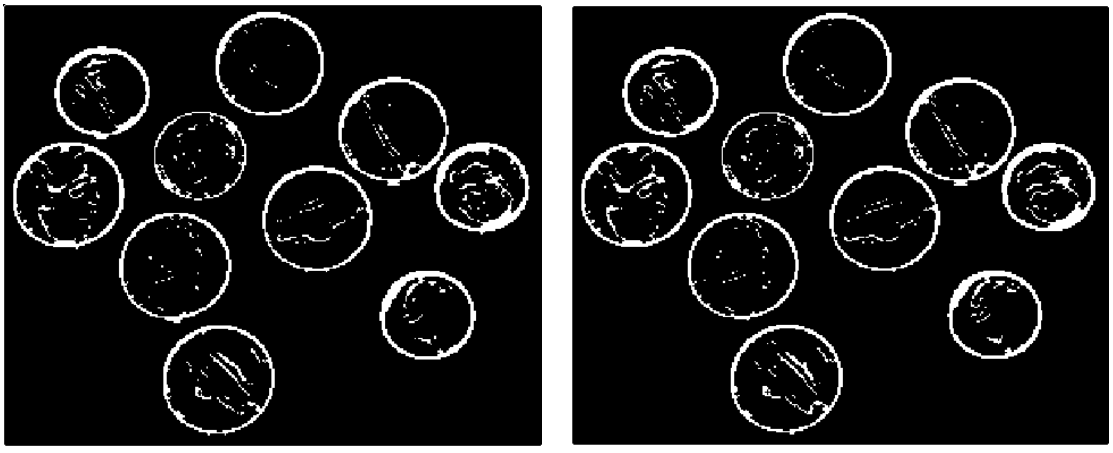


Figure 4.34: Edges obtained by: A. Simulation (left) B. Hybrid approach (right)

Three representative image test cases: *Lena*, *Cameraman* and *Coins* from the standard image processing test suite are considered in this experiment. These images are

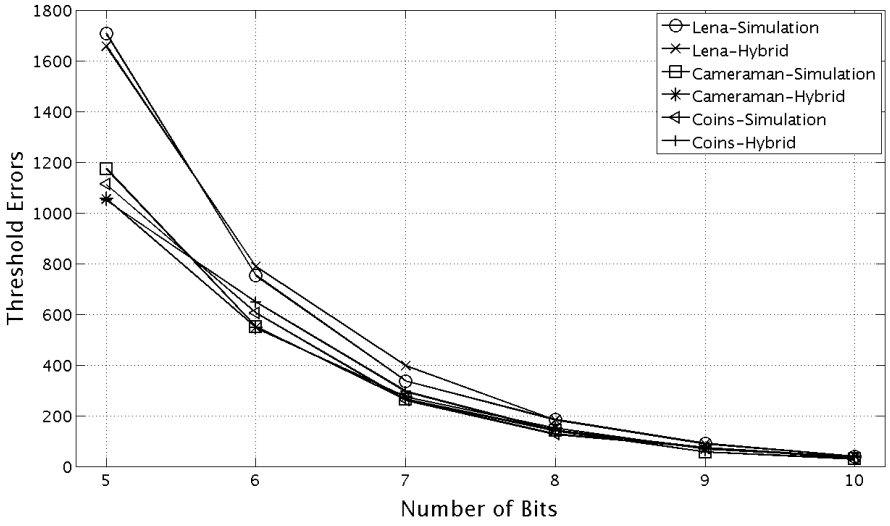


Figure 4.35: Edge detection: errors comparison

filtered through a Gaussian low-pass filter to emulate the blurring effect. The blurred image of *Coins* and its edges detected by using the edge detection considered in this example is shown in Figure 4.33. The edges of the *Coins* image detected using same schema which uses fixed-precision arithmetic with 5 bits in precision and the corresponding hybrid technique are shown in Figure 4.34. By visual inspection of the edges, it is clear that the edge quality of these images obtained in both cases are close. In order to have a better understanding, the plots in Figure 4.35 show the number of errors occurring after thresholding in comparison with the double precision case for both fixed-point and the proposed hybrid technique. The number of errors in both cases are of the same order with very little difference for various fixed-point precisions. Thus validating the statistical equivalence of the hybrid approach with fixed-point simulation for the case of this experiment.

Figure 4.36 shows the improvement factor obtained for various test cases and different levels of quantization of the unit normalized image input signal. The maximum relative error between the results obtained by fixed-point and *Hybrid* simulation is about 6%. As the number of precision bits increase, the image representation tends to be closer to the double precision case. Therefore, the improvement factor increases by several orders of magnitude with increase in precision. In this experiment, the improvement factor obtained indicate a speedup of three orders of magnitude.

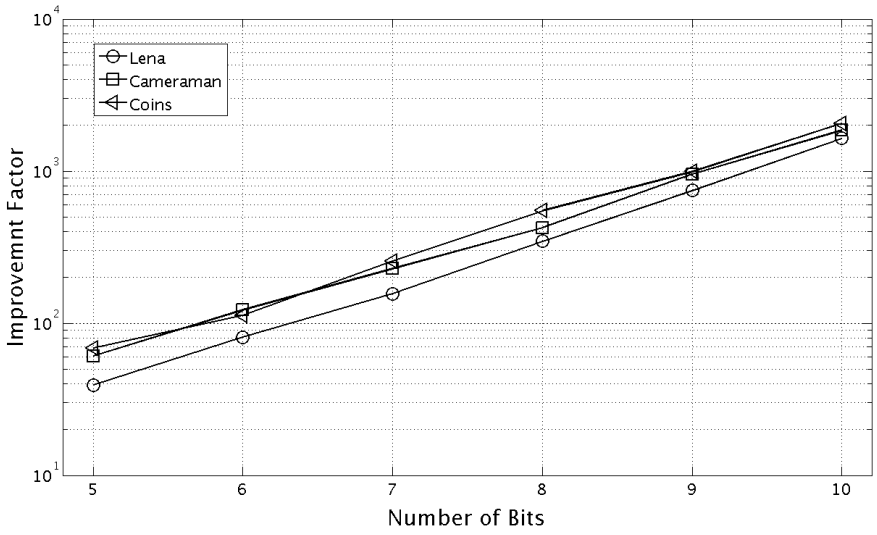


Figure 4.36: Edge detection: improvement factor (IF)

4.9.4 MIMO decoding with SSFE

MIMO[1] system for wireless communication promises improved efficiency communication efficiency [106] and has been widely studied and used in practical systems. The *SSFE*[2] [76] algorithm is a sphere decoding technique used for equalization of the received symbol and signal detection thereof. This technique belongs to a larger class of *Successive Interference Cancellation* techniques.

The schematic in Figure 4.37 shows the data-flow of the SSFE algorithm and its building blocks for a 4-transmit and 4-receive antenna case. The schematic is essentially performing the inversion of the R matrix obtained by performing the QR-decomposition of the channel matrix. As many smooth operation clusters as the number of receive antennas (one cluster corresponding to each antenna) along with those many QAM slicers numbered 4 down to 1 are shown in the schematic. This algorithm presents a case where un-smooth errors participate in computations and affect the output of other un-smooth operators. By changing the configuration of the algorithm, it is possible to vary the number of un-smooth operators.

The SSFE algorithm can be used in different configurations depending upon the number of constellation points explored at the output of each decision. The [4221] configuration is shown in Figure 4.38 with every number in the array indicating the number of constellation points explored at the output of every decision block corresponding to slicers sl_1 to sl_4. That is, at the output of the slicer sl_4, four neighbouring constellations are explored and hence there are four branches in the corresponding SSFE tree topology. Similarly, two neighbouring symbols are explored at the output of slicer sl_3

[1] Multiple Input Multiple Output Antenna
[2] Selective Spanning with Fast Enumeration

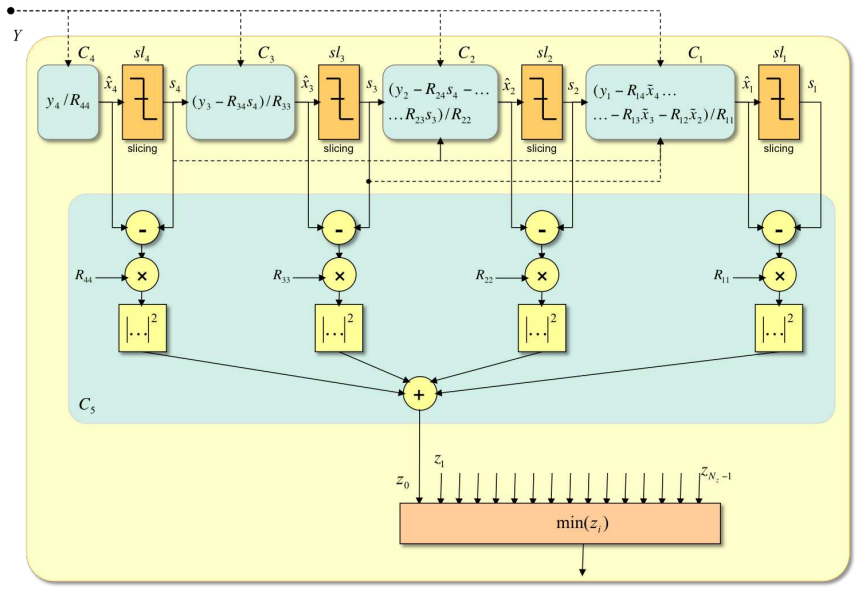


Figure 4.37: SSFE data flow model and associated smooth clusters

is indicated by two edges from each node corresponding to the slicer nodes sl_3 in the SSFE tree diagram. The other two slicers explore only one symbol and therefore it is indicated by just one edge in the SSFE tree diagram.

Consider the application of the proposed hybrid technique on this experiment. The smooth clusters are easily identified and the SNS model is derived in each case. Due to the absence of any memory elements within the smooth clusters, the power spectral density of the SNS models are white and are shaped by the number of errors added. There is just one operation in the smooth cluster corresponding to antenna 4 and hence the quantization noise PDF is uniformly distributed. The smooth cluster corresponding to antenna 1 on the other hand has 12 operators and hence can potentially consist of as many noise sources. Depending upon their relative powers, the noise shape is determined by matching the Kurtosis value and noise power using the technique described in Chapter 3. A double-precision simulation is carried out on all input test cases and the values at the input of every un-smooth operator is stored (i.e. $\hat{x}_4$, $\hat{x}_3$, $\hat{x}_2$ and $\hat{x}_1$ in this case) for use with the SNS model during hybrid simulation.

Figure 4.39 shows the degradation of BER[1] with decreasing channel noise in case of double precision simulation, fixed-point simulation and the hybrid technique for the [4 2 2 1] SSFE configuration. While it is expected that the fixed-point BER is inferior to the ones obtained by double precision, it can be seen that the difference between the BER obtained in cases of fixed-point simulation and hybrid technique is negligibly small. Two uniform precision assignments of 8 bits and 10 bits are considered for comparison purposes. The data obtained indicate that the error is only as large as 10%

[1]Bit Error Rate

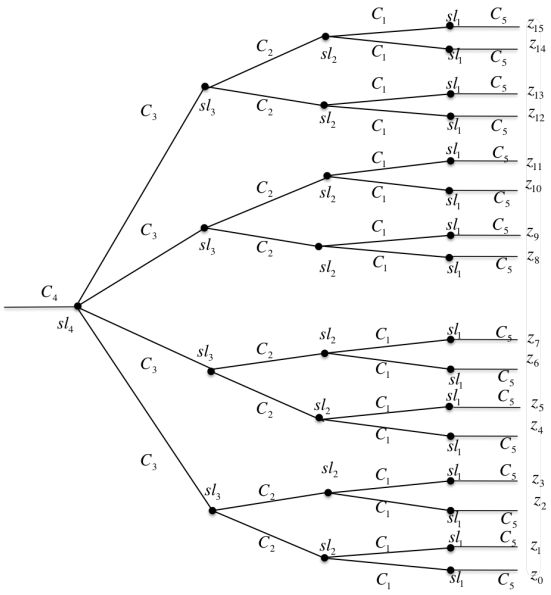


Figure 4.38: SSFE data flow with configuration [4221]

even in case of 8-bit (lower precision) precision assignment.

The improvement factor is dependent on the number of un-smooth (QAM decision) errors and hence the amount of noise in the system. The improvement factor in terms of performance evaluation time which as a function of the channel SNR is plotted in Figure 4.40 for three different precision assignments: 5-bits, 8-bits and 10-bits. It is seen that the improvement factor (IF) increases with reduction in channel noise and quantization noise. The increasing trend on the log scale as seen in Figure 4.40 is an indicator of the improvement that can be obtained in case of low BER simulations. The improvement factor achieved is as high as three orders of magnitude under low noise conditions.

The number of un-smooth errors is also correlated to the number of un-smooth operators. When there are many un-smooth operators, the number of instances where un-smooth error can actually occur also increases contributing to the overall increase in the number of un-smooth errors. Three configurations of the SSFE algorithm are considered to study the effect of varying the number of un-smooth operators with 14-bit precision fixed-point numbers. The results are summarized in the Table 4.1. The time taken (in number of seconds) for fixed-point and *Hybrid* simulation approaches are marked in the FP and Hy columns respectively.

Comparing the three configurations, the number of un-smooth operators at the antenna 4 remains a constant but are varied for other antennas. The total number of un-smooth operators for every configuration is shown in Table 4.2. With increase in the number of un-smooth operators, the number of smooth clusters also increase. This clearly causes an increase in the time required for fixed-point simulation. However, this also means that there are more points where un-smooth errors can occur and results in

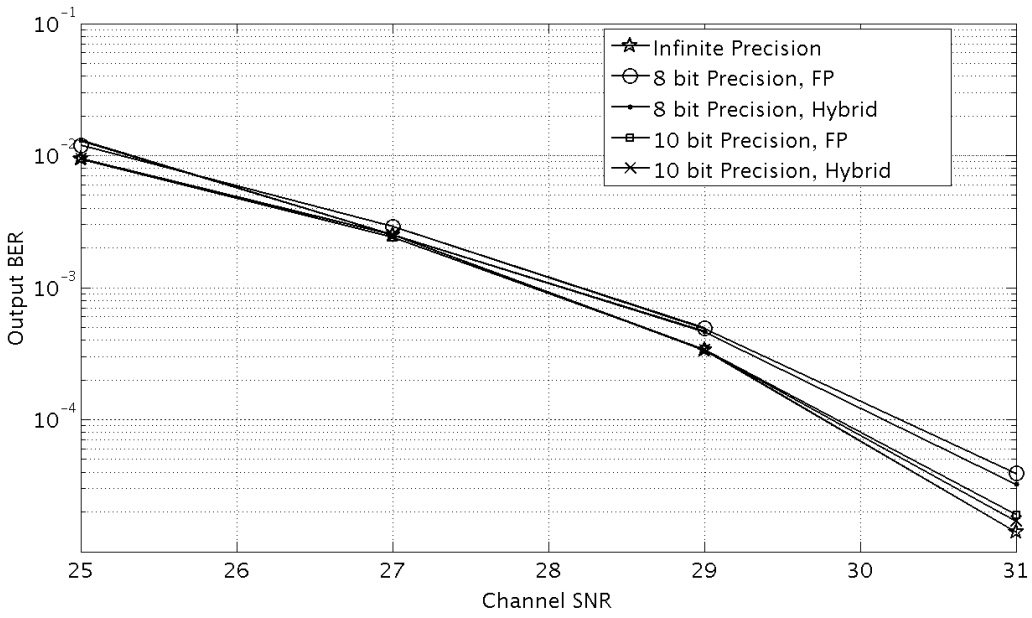


Figure 4.39: BER degradation in SSFE for configuration [4 2 2 1]

SNR	$m = [1, 2, 2, 4]$			$m = [1, 1, 2, 4]$			$m = [1, 1, 1, 4]$		
	FP	Hy	IF	FP	Hy	IF	FP	Hy	IF
19 dB	128	110e-3	**1.1e+3**	113	72e-3	**1.5e+3**	98	54e-3	**1.8e+3**
21 dB	128	96e-3	**1.3e+3**	113	72e-3	**1.5e+3**	98	39.e-3	**2.5e+3**
23 dB	129	98e-3	**1.3e+3**	113	67e-3	**1.6e+3**	102	31e-3	**3.2e+3**
25 dB	128	97e-3	**1.3e+3**	113	67e-3	**1.6e+3**	98	32e-3	**3.0e+3**

Table 4.1: Comparative study of execution times for different SSFE configurations

increased time for performing *Hybrid* simulation. Thus, although the numerator of the improvement factor in Equation 4.38 increases, there is also a corresponding increase in the denominator.

The results shown in Table 4.1 indicated that the relative increase in the denominator is greater than in the relative increase in the numerator of the expression for improvement factor in Equation 4.38. Therefore, while the consistency in increasing trend in the improvement factor across many configurations reinforces the rational behind using the hybrid approach, it is observed that the improvement factor decreases with increasing number of un-smooth operators.

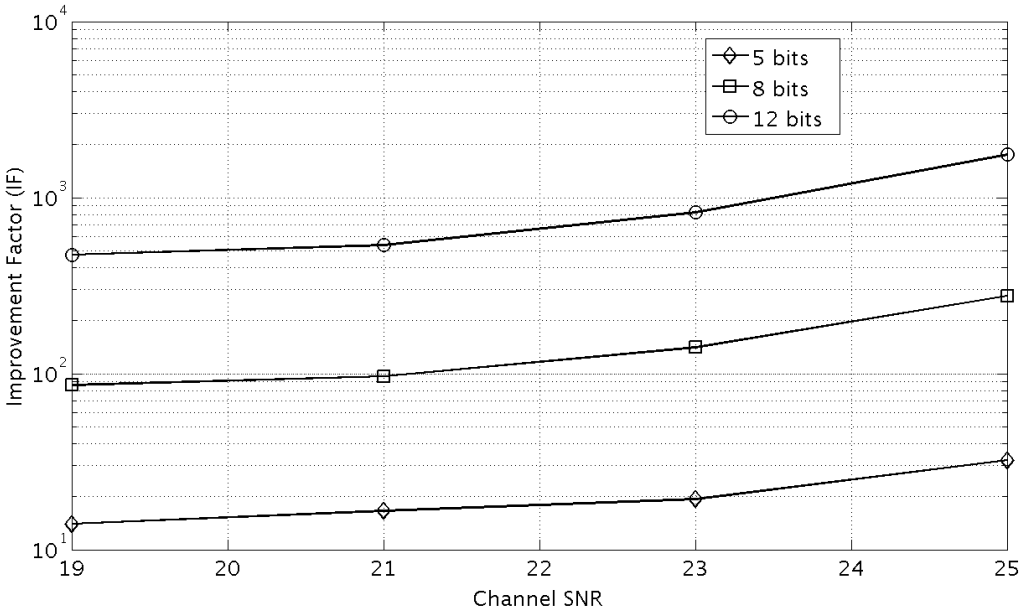


Figure 4.40: Improvement factor for SSFE configuration [4 2 2 1]

Configuration	Antenna 4	Antenna 3	Antenna 2	Antenna 1	Total
$m = [1, 2, 2, 4]$	4	2	2	1	64
$m = [1, 1, 2, 4]$	4	2	1	1	32
$m = [1, 1, 1, 4]$	4	1	1	1	16

Table 4.2: Number of Un-smooth operators in SSFE

4.10 Summary

The problem of evaluating the loss in performance of fixed-point systems when un-smooth operators are used for implementing signal processing systems is the focus of this chapter. Un-smooth operators are the ones whose response to fixed-point noise perturbation cannot be captured by the analytical techniques discussed in Chapter 2.

This chapter begins with a formal definition of un-smooth operators. Based on this definition, an algorithm to classify given quantizers into smooth or un-smooth is proposed. This algorithm is based on deducing the characteristic functions of the signals and requires just one high-precision simulation for estimation of signal statistics. As a second contribution in this chapter, a technique for accurate estimation of errors due to un-smooth quantization is developed. This technique is applicable in case of quantizers of any step-size. Though accurate and complete, the complexity of this technique is high and often requires efficient implementation of multi-dimensional numerical integration for its effective use in reducing the time. One drawback in this technique is that it is not applicable on systems with un-smooth operators in its feed-back loop.

As the third contribution in this chapter, a hybrid technique for accelerated simulation of fixed-point system is proposed. This technique makes use of the classification of operators as smooth or un-smooth and uses the analytical SNS model described in Chapter 3 to evaluate the impact of finite precision on smooth operators while performing simulation of the un-smooth operators during fixed-point simulation. In other words, parts of the system are selective simulated only in cases when un-smooth errors occur. The response to quantization noise follows analytical models otherwise. Thus, the effort for fixed-point simulation is greatly reduced.

The hybrid technique comes with a pre-processing overhead of identifying the un-smooth operators, identifying smooth clusters and deriving the SNS model. This overhead is a one time effort and can be used repeatedly for any assigned signal precision. This pre-processing overhead is often small in comparison to the time required for fixed-point simulation. The use of simulation for propagation of errors across simulation does not require the user to understand or characterise the non-linearities associated with un-smooth operators. The hybrid technique accelerates the process of fixed-point simulation by selectively applying the single-noise-source model. This algorithm is capable of working with topologies with feed-back, unlike the completely analytical technique. Several examples, from general signal processing, communication and image processing domains are considered for evaluation of the proposed hybrid technique. The acceleration in time taken measured in terms of improvement factor is seen to be orders of magnitude lower than the time required for fixed-point simulation.

The use of this technique for performance evaluatoin during word-length optimization has bearings on the choice of word-length optimization heuristic. Since the improvement factor is higher when high precision fixed-point word-length are used, it is better to choose the heuristic where the word-lengths descend from a high value in successive iteration. This is in contrast with the heuristic which suggests the ascent of word-lengths where the quantization noise remains high during many iterations. This results in smaller improvement factors and therefore does not take full benefit of the proposed *Hybrid* simulation approach.

Chapter 5

Hierarchical Word-length Optimization

Fixed-point refinement of signal processing systems presents a vast combinatorial optimization space that needs to be explored before arriving at optimal word-lengths. The solution space is known to increase exponentially with growing complexity of signal processing systems. A canonical representation of the word-length optimization algorithm was introduced in Chapter 1. The NP-hard nature [34] of the word-length optimization problem makes it difficult to guarantee optimality unless the entire search space is explored.

The use of heuristics has been the primary technique for solving the problem thus far. These heuristics include greedy approaches [18] such as the popular *Min* +1 *bit* and *Max* −1 *bit* techniques. Genetic algorithms and simulated annealing approaches have also been experimented [2; 3]. In [21], an analysis of the gradient-based greedy approaches is made by applying the popular *Lagrange Multipliers*. The *Marginal Analysis* technique based on this analysis has been proposed for word-length optimization. This technique is similar to the classical *Min* +1 *bit* algorithm except that the precision bits are incremented starting from 0 bits instead of minimum number of bits. In case of all such heuristic driven algorithms, increase in the number of variables is known to cause an increase in the time taken together with risks of reduced chances of finding the optimal solution. Therefore, this problem is more visible when complex systems with a large number of variables participate in the word-length optimization problem. Scalability aspects of the optimization algorithms is one of the important considerations in this thesis.

In this chapter, the first contribution addresses the complexity aspect by proposing a divide and conquer strategy to solve the word-length optimization problem in Section 5.1. This section describes the constraints on hierarchical decomposition of the system and techniques to decompose the word-length optimization problem according to the sub-system decomposition in order to reduce the complexity of large word-length optimization problems. An adaptation of the popular greedy *Min* +1 *bit* algorithm approach is proposed for solving the hierarchical word-length optimization

problem.

The second contribution addresses the combinatorial nature of the word-length optimization problem. In Section 5.4, an alternate convex optimization framework for word-length optimization is proposed. This technique uses the quantization noise-power instead of the number of bits as the optimization variable. The choice of using noise-power as the optimization variable helps in relaxing the integer nature of bit widths and uses an approximate, yet continuous Pareto-front for making the cost-precision trade-off. The quantization *noise-budgeting* problem which happens to be convex is formulated by applying the integer relaxation of the variable is proposed. This problem is equivalent to the word-length optimization problem in the sense that the solution to this problem is indicative of the optimal solution to the original word-length optimization problem. A standard convex optimization solver is used to solve the *noise-budgeting* problem to obtain optimality. In Section 5.4.5, the result obtained from solving the *noise-budgeting* problem is used to realise the fixed-point word-lengths that generates as much quantization noise as it is budgeted.

5.1 Divide and Conquer Approach

The classical word-length optimization problem described in Chapter 1 is combinatorial in nature. Although this problem is solvable when the problem size is small by an exhaustive search, the search space is known to grow exponentially with the addition of every new optimization variable. The divide-and-conquer approach proposed in this thesis aims at reducing the complexity of this problem.

The proposed divide-and-conquer approach essentially consists of three steps: dividing the given problem into smaller sub-problems, solving the simplified problem and combining the solution in order to solve the original bigger problem. The hierarchical decomposition of the given optimization problem is dealt with in Section 3.1. Classical word-length optimization approaches is used for solving the smaller problems. Finally, the hierarchical word-length optimization algorithm is proposed to perform the combination step.

5.1.1 Problem Division

It is clear from the discussion in the chapter 3 that the total noise power at the output of a system is given as a function of all sub-system output noise powers. The use of single-noise-source (SNS) model and the noise power contribution at various sub-system output is therefore sufficient to evaluate degradation in accuracy due to fixed-precision operations. Given the SNS model for all sub-systems, it is easy to compute the output quantization noise power. Also, the total cost of the system which is obtained by adding up the individual sub-system costs can be expressed as a function of quantization noise-power. Therefore, quantization noise-power is chosen as the optimization variable for solving the hierarchical word-length optimization problem.

Using quantization noise power at the sub-system output as an optimization variable, the cost minimization problem can be stated as

$$\text{minimize}\,(C\,(\mathbf{p}))\ \text{ subject to }\quad \lambda\,(\mathbf{p}) \leq \lambda_{obj}, \tag{5.1}$$

where $\mathbf{p}$ is a vector of total noise power at the output of each sub-system. Applying Equation 5.1 recursively starting from the topmost level to all levels in the sub-system hierarchy, it is possible to divide the global optimization problem into smaller and more manageable sub-problems.

Consider the hierarchical decomposition of the word-length optimization problem for the system shown in Figure 5.1. It consists of many sub-systems that have been decomposed into three levels of sub-system hierarchy. The sub-system components and their topological interconnect is also shown in Figure 5.1. The hierarchical decomposition of large systems is influenced by three factors which includes considerations of complexity, presence of un-smooth operators and architectural choice.

Presence of un-smooth operators

The first consideration arises from the fact that the use of quantization noise-power as an accuracy evaluation metric is not always possible. In the presence of un-smooth operators, it is difficult to continue using the quantization noise power metric to propagate the impact of fixed-point operations on the system output. Consequently, the global optimization problem cannot be expressed as a cost minimization problem subject to performance constraints as a function of sub-system quantization noise-power as given in Equation 5.1. Therefore, the sub-systems in a hierarchically decomposed system may not contain an un-smooth operator.

In the example of the system in Figure 5.1, the first sub-system level comprises of the four sub-systems separated by un-smooth operators as shown in Figure 5.2.

Number of optimization variables

Hierarchical decomposition of a signal processing system was discussed in Chapter 3 in Section 3.1. The large system under consideration is broken down hierarchically into a number of sub-systems as long as the complexity of individual sub-systems is not manageable. In the context of this chapter, each node corresponding to a sub-system is an optimization problem which is essentially a sub-problem of the global optimization problem. The number of edges fanning out of a node is the number of variables the optimization problem corresponding to that node has to deal with. Therefore, the hierarchical decomposition of large systems tends to be influenced by the complexity and size of individual systems.

In the example of the system in Figure 5.1, large sub-systems: S_1^1 and S_1^3 are further sub-divided into many smaller sub-systems. Here, S_i^j represents the j^{th} sub-system problem defined at the i^{th} hierarchical level. The levels pf hierarchy begins with 0 which represnts the entire system. In the second level, the sub-system S_2^4 is decomposed to obtained the sub-systems in the third level. Whereas, small sub-systems

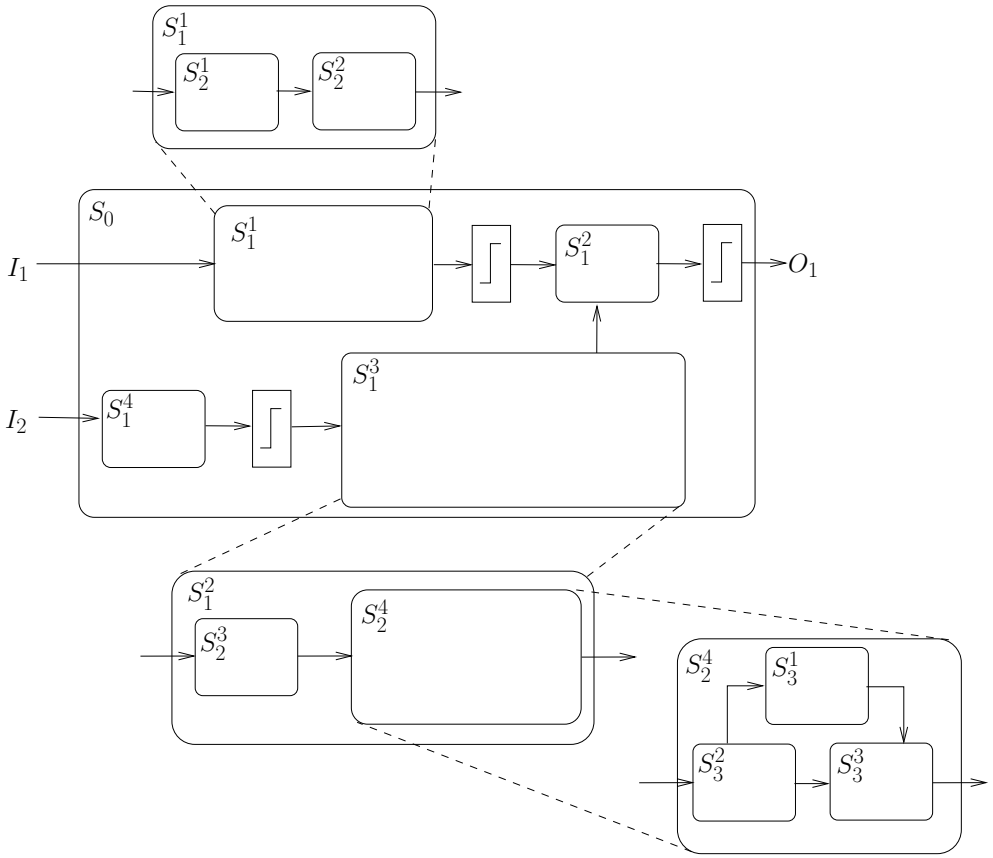


Figure 5.1: System Level Hierarchical Decomposition for Word-length Optimization

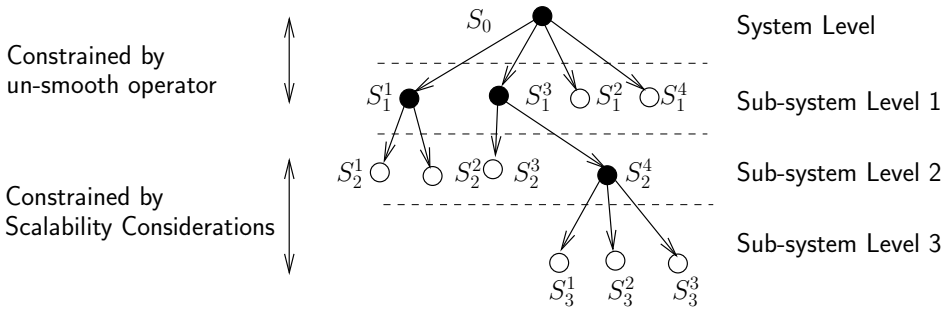


Figure 5.2: Word-length Optimization Problem: division into sub-problems

such as S_1^2 and S_2^3 are small enough and are not decomposed further. Each of these nodes in Figure 5.2 represent a smaller word-length optimization problem.

In this example, the performance evaluation function $\lambda_(\mathbf{p})$ evaluates the system output accuracy performance metric which is essentially not quantization noise power. The vector $\mathbf{p} = [p_1^1, p_1^2, p_1^3, p_1^4]$ is the output quantization noise of the sub-systems

$S_1^1, S_1^2, S_1^3, S_1^4$ respectively. Here, for the sake of simplicity in illustration, there is just one fan-out from every sub-system. This need not be true in general. When there are multiple fan-outs from a sub-system, the quantization noise at each of the outputs will have to be considered and hence an element corresponding to each of the outputs is added into the vector $\mathbf{p}$. The global word-length optimization problem is obtained by using the appropriate cost function $C(\mathbf{p})$ and the fixed-point accuracy evaluation function $\lambda(\mathbf{p})$ into Equation 5.1. Similarly, the nodes in lower hierarchical levels represent the decomposition of the sub-problems into progressively smaller sub-problems of word-length optimization. When each of these problems is individually solved and their solution combined, the global system level problem is also solved. The solution thus obtained need not be the most optimal in general as nothing can be conclusively said about the quality of the leaf level problems themselves. Various factors including the way a sub-system is defined can have an impact on the final solution.

Architectural Choices: Resource Sharing

A third constraint is that of the implementation architecture. In the design of a micro-architecture, resource sharing refers to the fact that a hardware primitive such as an adder or a multiplier is shared to perform many computations to implement a given signal processing algorithm. The degree of freedom to choose between different fixed-point word-lengths for executing an operation heavily depends on this choice. An optimization problem has as many degrees of freedom as the number of variables participating in the optimization. Two paradigms for word-length optimization: i) uniform word-length assignment, ii) multiple word-length assignment discussed in Chapter 2 is worth recalling here. The uniform word-length optimization problem has only one degree of freedom as all the operations have a single fixed-point word-length format assigned. On the other extreme, the multiple word-length optimization problem has as many degrees as the number of operations in the signal processing algorithm. As the resources are shared, say for example there happens to be just one binary adder and one binary multiplier for the implementation of the algorithm, the problem of word-length optimization for a system with such constraints can have a maximum of 6 variables corresponding to two inputs and one output of the multiplier and adder. Similarly, if there are two types of binary adders and two binary multipliers with different fixed-point configurations, there can be 12 variables. Essentially, resource sharing reduces the degrees of freedom available for the optimization problem. More sharing leads to lesser number of variables in the optimization problem and vice-versa.

Complex signal processing systems are typically built using standard signal processing primitives such as the Fast-Fourier-Transform (FFT) algorithm, filters and equalizers. Some practical scenarios such as the use of FFT algorithm in a MIMO-OFDM transceiver design, the use of FIR filters for discrete-time wavelet-based dyadic decomposition and reconstruction of images can be found in practice. Many architectural choices exist for implementing a given signal processing algorithm. One of the possibilities is to use an instance of the block primitive and all operations are executed on that one instance whenever it requires to be executed. Other possibility includes

having more than one instance of the primitive to perform the computations in parallel. In general, there can be as many as the parallelism of the signal processing algorithm allows. An extreme possibility is that each instance of the algorithm representing more than one sub-systems in the signal processing algorithm has a dedicated computational resource. This is referred to as the full parallel hardware architecture in this thesis. Architectural decision can be based on various other factors and it is not the focus of this thesis.

Resource sharing in the context of hierarchical decomposition of systems occurs when one instance of an algorithm such as a filter or an FFT is used for implementation of various sub-system blocks of the system. The implementation of the algorithm may be in hardware or software. If it is a hardware implementation, it is used in a time-multiplexed fashion whenever it is called. If it is a software implementation, usually it is implemented as a function call. Then, the data-types of the variables used for calling the function and local variables within the function are implemented are fixed. Like in the case of operator level resource sharing, the functionality of the system is not altered by sharing the same fixed-point implementation of the algorithm. It is assumed that such constraints are imposed by the designer and that; when an implementation is shared across two sub-systems, they essentially use the same fixed-point implementation. Therefore, they essentially represent one set of optimization variables rather than two. In effect, sharing resources reduces the number of variables participating in the system.

In this section, the objective is to study the impact of resource sharing on the complexity of the word-length optimization problem. The hierarchical decomposition in Figure 5.2 assumes that there is no resource sharing between implementations of different sub-systems. Two scenarios can occur when two or more sub-systems participate in resource sharing. To illustrate these two scenarios, consider adding the following architectural resource sharing constraints.

- **Scenario 1:** Sub-systems S_3^2 and S_3^3 share the same primitive
- **Scenario 2:** Sub-systems S_3^1 and S_2^2 share the same primitive

The first scenario is that all the sub-systems sharing the resource are at the sub-system same level. In such cases, the number of optimization variables reduces by the extent of resource sharing. Consider the implementation of the sub-system S_2^4 in full parallel hardware. The node corresponding to this sub-system has three edges fanning out and hence, the word-length optimization problem to optimize the node corresponding to S_2^4 essentially consists of three optimization variables. If the first constraint constraint from the above list is considered, the number of variables reduces just to two. In other words, since the sub-systems S_3^2 and S_3^3 are essentially implemented using the same primitives and are optimized jointly, which effectively reduces the number of variables to two. The hierarchical graph transformation corresponding to this constraint is as shown in Figure 5.3.

The second scenario is when sub-systems that share resources are present in different hierarchical levels. Then, it affects the hierarchical decomposition of the system. Then,

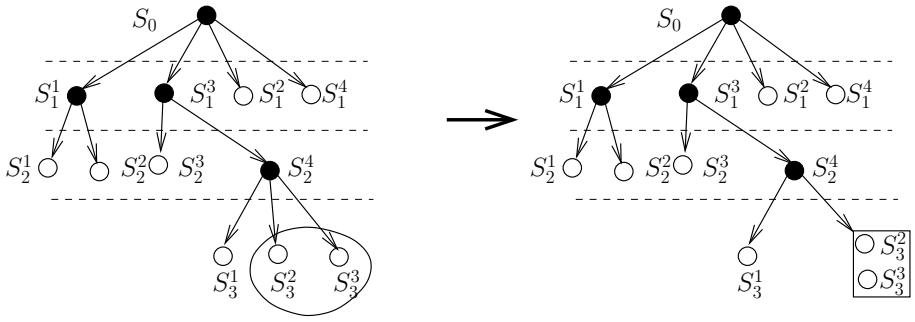


Figure 5.3: Resource sharing Scenario 1

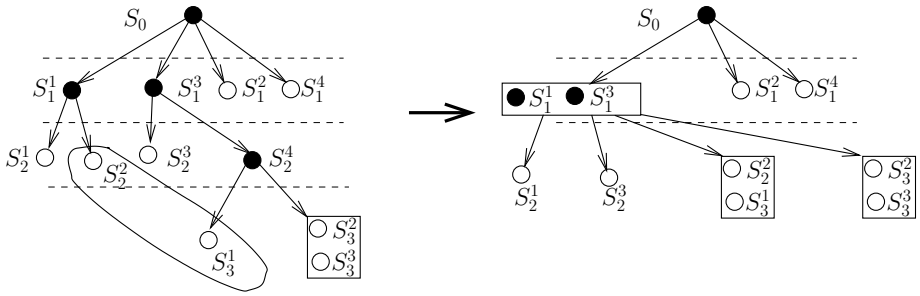


Figure 5.4: Resource sharing Scenario 2

the output noise-power contributed by the primitive participates in the optimization of more than one sub-system simultaneously. It is therefore not possible to keep the two optimization variables independent. Since the optimization problem they contribute are also different, the uniqueness of the implementation requires that the impact of fixed-point quantization of the primitive be taken care of simultaneously while solving both problems. Therefore, it is required to modify the hierarchy to reflect this consideration. In such cases, the hierarchy needs to be collapsed between the levels in which the sub-system are participating in resource sharing.

In case of the example in Figure 5.1, consider adding the second resource sharing constraint mentioned above. From Figure 5.3, S_3^1 and S_2^2 are at different levels and belong to different parents at the second level. When this resource sharing constraint is applied, the noise power output from sub-systems S_3^1 and S_2^2 have to be considered simultaneously. This is possible by collapsing the third level and replacing S_2^4 by nodes corresponding to its sub-problems. Since the parents of the combined node S_3^1 and S_2^2 are different in the original hierarchical decomposition, they also have to be merged. That is, nodes S_1^1 and S_1^3 also have to be merged. This transformation is as shown in Figure 5.4. It has to be noted here that it not only collapses the node S_2^4 but also does not let splitting of node S_1^1 and S_1^3.

In summary, having architectural resource sharing constraints can have a profound impact on the decomposition structure and the number of optimization variables at

each level in the hierarchy.

5.1.2 Conquering Sub-problems

The optimization problem can be recursively divided until the leaf-level problem size is reached. The size of a leaf-level optimization sub-problem is small enough to be solved by using classical word-length optimization technique. The minimum cost of the global system is achieved by ensuring minimum cost of each of the leaf-level sub-systems. The leaf-level sub-system optimization sub-problem can be written in the canonical form as

$$\min(C_i(\mathbf{w_{d_i}})) \quad \text{subject to} \quad \lambda_i(\mathbf{w_{d_i}}) < p_i, \tag{5.2}$$

where C_i and λ_i are the cost and performance functions of the i^{th} leaf-level sub-system. These are functions of the various word-lengths $\mathbf{w_{d_i}}$ assigned to the fixed-point operations in the i^{th} sub-system. p_i is the noise power budgeted assigned to the i^{th} sub-system.

In other words, implementation costs of the i^{th} sub-system is minimized under the accuracy constraint p_i. Essentially, the divide-and-conquer approach for solving the global optimization problem is a problem of optimal noise-power budgeting to each of the sub-system blocks such that the minimum performance criteria at the global system output is satisfied.

5.1.3 Combining to Solve the Global Problem

In the proposed divide-and-conquer approach, the solution to the word-length optimization problem is constructed in a bottom-up fashion. The optimal noise power at the output of every sub-system evolves over the course of the word-length optimization iteratively. In every iteration, a sub-system noise-power is assigned to each of the sub-systems and all the sub-system optimization problem as described in Equation 5.2 is solved.

The guiding philosophy behind the proposed hierarchical optimization approach is to optimize the sub-systems contained within the system in order to optimize the system in totality. In this section, an algorithm that outlines the contours of an iterative hierarchical optimization technique is presented. This algorithm is suitable for use in solving the hierarchical word-length optimization problem iteratively and independent of the actual heuristic used for driving the optimization. The various steps involved in performing word-length optimization using the divide and conquer approach explained in the previous sections are given in the algorithm outlined in Procedure 5.1 and 5.1a.

The function *GetHierarchicalSystemTree()* returns the tree $T(V, E)$ by hierarchically decomposing the system level word-length optimization problem. In case of the example show in Figure 5.1, the tree in Figure 5.4 is returned. Each node in V in the tree corresponds to a sub-system optimization problem. The edges E denote the parent-child relationship between the nodes in the tree and always point from the parent node to one of the child nodes. If there are no out-edges from a node V_p, it represents a leaf

level sub-system optimization problem. The leaf level sub-system optimization problem is intended to be solved by using classical word-length optimization techniques. On the other hand, a node which is not at the leaf level is always connected by edges in E to its children nodes. The number of edges going out from a node is the total number of optimization variables that participate in the optimization problem corresponding to that node.

Indeed, the benefit of the hierarchical decomposition is best obtained by wisely choosing the sub-system boundaries. In this thesis, the proposed technique for hierarchical word-length optimization is the first step in this direction. The idea is to be able to apply the proposed technique to any given hierarchical decomposition of the optimization problem defined by the user.

The hierarchical word-length optimization tree is rooted at V_R: the node corresponding to the system-level noise budgeting problem. Starting from the root node V_R, the tree branches out progressively into smaller problems of lesser complexity and eventually to the leaf-level problems whose complexity is small enough to be managed by classical word-length optimization techniques. The root of the tree V_R is obtained by calling the function *GetTreeRoot()*. The optimization process is iterative in nature and it requires the previous states of the optimization process to be saved in order to guide the decisions taken during subsequent iterations. The previous states consist of the cost and the quantization noise power of the sub-system variables participating in the optimization problem. For example, in the example shown in Figure 5.4, the global problem S_0 has three sub-systems S_1^4, S_1^2 and the composite sub-system (S_1^1, S_1^3). Hence, three cost and performance parameters define the state of the optimization problem. Likewise, the problem corresponding to the composite system (S_1^1, S_1^3) has four sub-systems, And knowing the output noise power and cost of these sub-systems is sufficient to define the optimization state of the composite sub-system (S_1^1, S_1^3). In case of the leaf-level sub-system optimization problem such as S_2^3, the classical word-length optimization is used and hence the word-lengths of each of the fixed-point operations and the associated cost define their optimization state. To begin with, there is no previous state for any of the optimization problems as their optimization has not been attempted. In other words, none of the nodes in the tree have been visited and the noise-power at the output of any of the sub-system is unknown. The procedure *OptimizeSubsystems()* is first called with the root node V_R, the hierarchically decomposed optimization problem tree $T(V, E)$ and the system level accuracy performance metric λ_{global}.

Procedure 5.1 : Divide-and-Conquer Word-length Optimization

1: * *Decompose the system-level optimization problem into a hierarchy of sub-problems* *\
2: $T(V, E)$ = GetHierarchialSystemTree();
3: V_R = GetTreeRoot($T(V, E)$)
4: * *Begin with the global quantization noise budgeting problem* *\
5: [Cost, Performance] = OptimizeSubsystems(V_R, $T(V, E)$, λ_{obj});

Procedure 5.1a : OptimizeSubsystems(V_r,Tree,λ_{target})

```
\* Check if this node has previous states *\
if notVisited(V_r) then
  [p̂] = GetInitialPowerBudget(V_r, Tree, λ_target);
  for all Nodes V_d where Edge: V_r ↦ V_d ∈ E do
    if isLeafNode(V_d) == true then
      \* V_d is small enough to be optimized *\
      [c_d, p_d] = ClassicalWLOpt(V_d,p̂_d);
    else
      \* Recursive call to optimize node V_d by optimizing its sub-systems *\
      [c_d, p_d] = OptimizeSubsystems(V_d,T(V,E),p̂_d);
    end if
  end for
  p = [p_1 ... p_n];   \* Say, there are n children: V_d nodes *\
  c = [c_1 ... c_n];
  \* Store the state for use in next iterations *\
  V_r.state = [c, p];
else
  \* Carry forward previous state from the previous optimization attempt *\
  [c, p] = GetState(V_r);
end if
C_total = GetTotalCost(c);
λ_total = GetTotalPerformance(p);
\* Visit each child V_d of V_r in order of signal flow precedence *\
while [λ_total, C_total] is not an acceptable trade-off point do
  \* Continue with states in the previous iteration *\
  [c, p] = GetState(V_r);
  \* Estimate new power budgets *\
  [p̂] = RefineHeuristicsPowerBudget(V_r, Tree, P_target, p, c);
  for all Nodes V_d where Edge: V_r ↦ V_d ∈ E do
    if isLeafNode(V_d) == true then
      \* V_d is small enough to be optimized *\
      [c_d, p_d] = ClassicalWLOpt(V_d,p̂_d);
    else
      \* Recursive call to optimize node V_d by optimizing its sub-systems *\
      [c_d, p_d] = OptimizeSubsystems(V_d,T(V,E),p̂_d);
    end if
  end for
  p = [p_1 ... p_n];   \* Say, there are n children: V_d nodes *\
  c = [c_1 ... c_n];
  V_r.state = [c, p];
  C_total = GetTotalCost(c);
  λ_total = GetTotalPerformance(p);
end while
return [C_total, λ_total]
```

Procedure 5.1a traverses the tree $T(V, E)$ recursively starting from the root node: from which it is first invoked in Procedure 5.1. The first step in this procedure is to check if a history is available for the optimization problem corresponding to node V_r. In other words, if it is being visited the first time. In case the node is being visited for the first time, an initial noise-power budget $\hat{\mathbf{p}}$ is assigned to the sub-systems. The technique used to make this assignment is a part of the heuristic used for solving the word-length optimization problem. The corresponding sub-system cost is calculated in case of every sub-system by either recursively calling the *OptimizeSubsystems()* or by calling the classical word-length optimization algorithm function *ClassicalWLOpt()*. The resulting noise power vector $\mathbf{p}$ and sub-system cost vector $\mathbf{c}$ is recorded as the state of the optimization problem corresponding to node V_r. If it is not the first call to optimize the problem corresponding to node V_r, the stored state is read back.

The total cost and performance of the optimization problem is evaluated using the sub-system cost and noise power vectors. While the present trade-off point for the optimization problem corresponding to the node V_r is not satisfactory, the noise-power budgets are refined and a new vector $\hat{\mathbf{p}}$ is determined. Once again, it is attempted to realise the newly assigned noise budget by optimizing its sub-systems. A new vector of noise power: $\mathbf{p}$ and sub-system cost $\mathbf{c}$ is obtained. The total cost and performance C_{total} and λ_{total} is evaluated and the new state of optimization is recorded. The whole process repeats in the while-loop as long the trade-off points attained is not satisfactory.

Storing the states of the optimization problem is critical in this process. If this state information is not stored, every call to *OptimizeSubsystems()* will have to perform optimization of each sub-system from the beginning. As the heuristic used or the step-size between iterations does not change, the optimization problem treads the same evolutionary path every time it is executed. Therefore, the state information is saved for every optimization problem in order to save time. This also helps store the fixed-point word-lengths. When the optimization iteration terminates, one has to just lookup the states of the leaf-level optimization problem to obtain the fixed-point formats of each operation.

5.2 Adapting a Greedy Algorithm

Two popular word-length optimization algorithms *Min* $+1$ *bit*, *Max* -1 *bit* also referred to as the *middle-ascent* and *steepest-descent* respectively, belong to the class of *greedy*, multi-variable optimization heuristics [18]. Both algorithms use the actual fixed-point precision word-length of various operators as optimization variables. They use an iterative procedure for word-length refinement which is similar to the framework outlined in Procedure 5.1a to refine the word-lengths.

In this thesis, the *Min* $+1$ *bit* algorithm is chosen for adaption to the hierarchical optimization technique. This is chosen owing to its simplicity and popularity. Since the noise power is used as the optimization variable instead of the number of precision bits, this algorithm is referred to as the *Max* $-\delta_P$ *dB* algorithm. Here, δ_P is the noise power step expressed in decibels and hence *"dB"* is used in the place of bits.

5.2.1 Initialization: maximal value of p_i

The initialization step defines *GetInitialPowerBudget()*, which assigns the maximum noise power value for each noise source p_i. The maximum initial noise power at the output of a sub-system is that value which can individually reach the total accuracy target independent of other noise sources. That is, for finding the maximal value of a variable p_i, the other noise sources $p_j, \forall j \neq i$ are set to zero. This is formally stated as

$$\max(p_i) \quad \text{such that} \quad \begin{matrix} p_j = 0 \quad \forall j \neq i \\ \lambda(\mathbf{p}) \geq \lambda_{target}, \end{matrix} \tag{5.3}$$

where λ_{target} is the maximum acceptable noise-power at the system output. The actual values of various P_i is different from one another depending upon the path function from a given sub-system.

5.2.2 Iterative Optimization

In every iteration of the while-loop in procedure *OptimizeSubsystems()*, the sub-system cost and performance are used to calculate the total system cost and performance. The call to *RefineHeuristicsPowerBudget()* in every iteration re-adjusts the noise power budgets.

To find the best direction for convergence onto the optimal noise power distribution of the variables, the opportunity to decrease the noise power of each variable P_i by a value δ_p is explored. Let $\mathbf{p_k}$ and $\mathbf{c_k}$ be the quantization noise power and cost vector obtained at the k^{th} iteration. Let $\delta_{\mathbf{p_i}}$ be a vector having all its element null except the element i which is equal to the noise step δ_p. The change in system performance at the k^{th} iteration $\Delta\lambda_i(\mathbf{p_k})$ is calculated by evaluating the different between the total system performance in the previous interaction and the total system performance due to the change in the sub-system noise-power assignment. This is written as

$$\Delta\lambda_i(\mathbf{p_k}) = \lambda(\mathbf{p_k} - \delta_{\mathbf{p_i}}) - \lambda(\mathbf{p_k}). \tag{5.4}$$

The corresponding change in cost $\Delta c_i(\mathbf{p}(\mathbf{k}))$ can be evaluated as a difference in the total system cost in the previous iteration and the total system cost resulting due to change in noise-power assignment. This is written as

$$\Delta c_i(\mathbf{p_k}) = c(\mathbf{p_k} - \delta_{\mathbf{p_i}}) - c(\mathbf{p_k}), \tag{5.5}$$

where κ_i, the gradient of change corresponding to each of the i^{th} sub-system is the ratio of the change in performance and the change in cost of the i^{th} sub-system.

$$\kappa_i(\mathbf{k}) = \frac{\Delta\lambda_i(\mathbf{p_k})}{\Delta c_i(\mathbf{p_k})}. \tag{5.6}$$

In every successive iteration, the overall system performance improves which means that the sub-system noise levels decreases while the cost of the total system increases. All the κ_i are sorted in descending order such that the first element in the sorted list

corresponds to that sub-system which improves the noise performance of the overall system at the smallest expense of the cost.

The step change in noise power δ_P is set to a convenient value by comparing the orders of magnitude of noise power allowed in the system. If the step is too small in magnitude, too many iterations are required whereas if it is too big, the solution obtained could be far from optimality.

5.2.3 Optimality of the Noise-Budgeting Technique

The hierarchical optimization strategy presented in the previous section essentially relies upon the ability of the classical word-length optimization technique. Therefore, it is first important to discuss the classical word-length optimization process before dwelling into the proposed hierarchical technique.

Given the NP-hard nature of the classical word-length optimization problem, none of the existing techniques can achieve true optimality of the word-length optimization choices. Therefore, it is expected that the hierarchical technique would at best be only as optimal as the classical optimization algorithm used to optimize its sub-systems. Moreover, the choice of the noise step δ_{pb}, the choice of initial power distribution and the heuristics used to refine the noise-power budgets in the proposed hierarchical method casts its effects on the result obtained.

In this section, the optimality of the greedy algorithms in general and the *Min* +1 *bit* algorithm in particular is considered. The combinatorial nature of the word-length optimization problem is well known. The hierarchical formulation of the optimization problem does not affect this nature.

Classical Word-Length Optimization

In the classical word-length optimization problem, word-lengths are used as optimization variables. The integer values of the fixed-point word-length discretizes the quantity of noise injected into the sub-system. The noise-power at the sub-system output, which is a function of the input noise-sources also takes on discrete values.

In the greedy heuristic such as the *Min* +1 *bit* algorithm, the initialization step is set to the minimum number of bits such that only the minimum accuracy criteria is met individually by each of the fixed-point variables. This is similar to Equation 5.3 described in the initialization condition in the *Max* -δ_{pb} algorithm. It is written as

$$\min(w_i) \quad \text{such that} \quad \begin{array}{l} w_j = max\_bits \quad \forall j \neq i \\ \lambda(\mathbf{w}) \geq \lambda_{target}, \end{array} \tag{5.7}$$

where *max_bits* is the maximum bits that can be assigned to the fixed-point numbers. This is usually dependent on the platform on which the system is implemented and typically the word size of 32 bits is used for this purpose. The word-lengths w_j of all signals thus obtained after initialization are the minimum fixed-point word-lengths assignable to each of the operations. The performance and cost of such a fixed-point

system are evaluated. Clearly, the minimum word-length vector $\mathbf{w}_{min}$ has the smallest cost but its accuracy is very compromised.

The objective of the *Min* +1 *bit* algorithm is to iteratively improve the precision of the fixed-point operations such that the required accuracy performance is satisfied. The iterative fixed-point refinement process comprising of cost and performance evaluation in every iteration is shown in Figure 5.5.

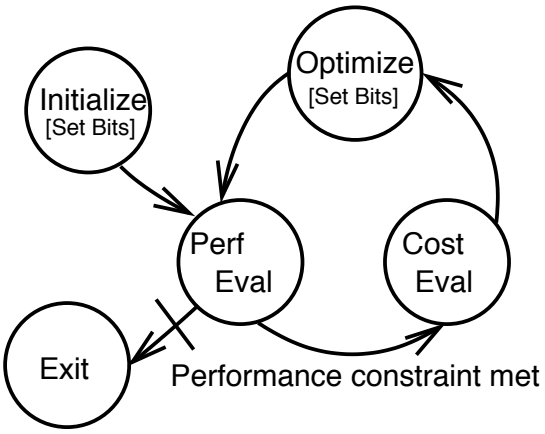


Figure 5.5: Iterative Fixed-point refinement

In every iteration, the performance and the cost of the fixed-point system with assigned fixed-point word-lengths are evaluated. The fixed-point operator which contributes minimal increase in the overall cost while giving the best performance is chosen. The accuracy performance of the system is expected to improve in every iteration until it satisfies the specified performance criteria. It has to be noted here that the exit condition is merely taking care of meeting the performance criteria. It can so happen that many bit-increments may result in the same performance and cost trade-offs. But choosing one over the other can indeed be sub-optimal for the scenarios presented in successive iterations. In such a circumstance, the *Min* +1 *bit* algorithm does not search all possible options but goes with just one of the available options without checking for its consequences in successive iterations. Therefore, by following the *Min* +1 *bit* algorithm it is not possible to guarantee that minimum cost is achieved. It also means that there is a chance that solution obtained is indeed optimal. However, nothing can be said about it unless all other options are explored. In other words, it is just that optimality cannot be proved even when an optimal solution is obtained.

The two-input butterfly computation structure commonly used in FFT computation is considered as an example to illustrate the effects of the *Min* +1*bit* heuristic. In particular, this experiment shows the impact of a bad decision which is made in the initialization step.

Figure 5.6 shows all possible cost and accuracy trade-off points with energy cost on the x-axis and the quantization noise power on the y-axis in the log-scale. Each of these points corresponds to the various cost vs. output noise-power trade-off for the two

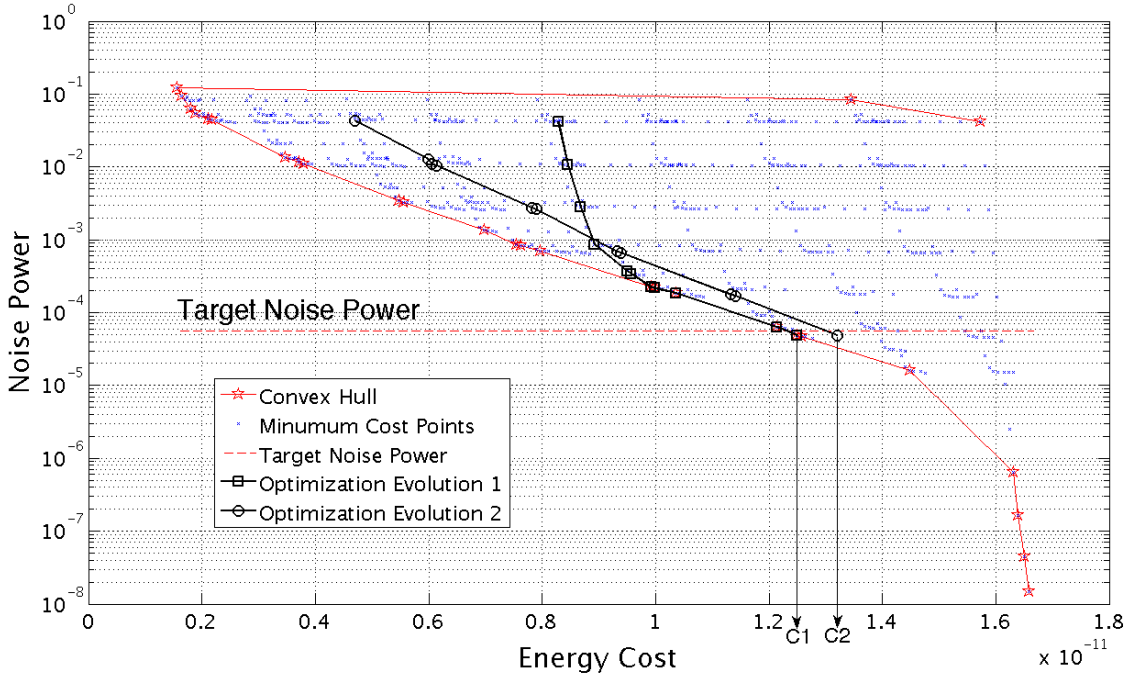


Figure 5.6: Cost vs. Accuracy trade-off for an FFT butterfly using *Min* +1 *bit* word-length optimization algorithm

input butterfly. The *Convex-Hull* boundary marked on the figure is a convex polygon surrounding all the trade-off points. The convex-polygon is obtained by considering the points on a linear scale. However, in order to distinctly show all the trade-off points, the graph is plotted on the semi-logarithmic scale with the vertical axis on the logarithmic scale. The choice of logarithmic scale is the reason why the polygon is not closed in Figure 5.6.

The optimal choice for any given accuracy constraint is obtained by choosing the points closest to the origin. The boundary of the *Convex-hull* nearest to the origin contain many such points and can hence be considered the Pareto-optimal front of the two-input butterfly.

The iterative refinement of the *Min* +1 *bit* algorithm is performed with two different starting points and the evolution of the solution is marked on the graph. The trajectory of evolution of the cost and noise-power at different operating points explored in every successive iteration is plotted. The heuristic used to initialize and optimize the initial word-length or noise power assignment governs the optimality of the result obtained. In the first case (optimization evolution 1), the evolutionary trajectory eventually converges with the Pareto-optimal boundary line and eventually satisfying the accuracy constraint. As the solution in this case lies along the Pareto-optimal curve, the cost C_1 is the smallest cost of the fixed-point implementation such that the input noise-power

constraint is satisfied. The second case (optimization evolution 2) shows that the evolution trajectory runs parallel to the Pareto-optimal curve. The cost $C2$ obtained in the second iteration is larger than cost $C1$ and is therefore clearly sub-optimal. This is in spite of the fact that the initial starting point in the second case was much closer to the Pareto-optimal curve than in the first case. In fact, the initialization as suggested by Equation 5.7 was performed in the second case.

This is an example of the un-guided behavior of the greedy algorithm. Clearly, the initial condition plays an important role in arriving at an optimal solution. It is not possible to determine the initial condition from where optimality can be guaranteed by following the Min $+1$ bit heuristic. In other words, starting from a point which may lead to optimality is a matter of chance. The reason for this kind of behavior is the non-convexity of the cost vs. fixed-point word-length trade-off and that the word-length optimization is an integer programming problem and the word-lengths cannot take real numbered values.

5.2.4 Hierarchical Optimization

The convergence of the loop onto the exit condition is decided by the kind of choices presented for carrying out the cost vs. accuracy trade-off in every iteration. In the hierarchical setup, as the smaller problems are solved, the cost-performance trade-off points of the sub-systems placed one level above the hierarchy are evaluated. The system level choices depends on the cost-performance trade-off analysis of the sub-systems. If the choice of word-lengths of the sub-systems is not optimal due to the use of classical word-length optimization for sub-system optimization, it wrongly influences the system level trade-off balance. The effect of sub-optimal choices can accumulate across various hierarchies and take the working point very far away from the optimal choice. Consequently, it is not necessary for the hierarchical and the classical flat word-length optimization routines generate the same result. However, the divide and conquer nature of the hierarchical procedure reduces the total number of global iterations. Thus reducing the number of iterations required for performing the optimization.

Combinatorics of the Noise-Budgeting Problem

The discrete combinatorial nature of the original word-length is the result of the integer word-length assignment. Although the hierarchical approach defined in Equation 5.1 introduces a real valued optimization variable: the sub-system quantization noise-power instead of integer word-lengths, the optimization problem continues to preserve the discrete combinatorial optimization nature of the original problem. This is because of the actual values of noise-power assigned to hierarchical optimization variables take on discrete values depending upon the word-length assignments. For example, consider the case of hierarchical decomposition of the FFT algorithm. If the two-input-two-output butterfly is considered to be a sub-system with which the FFT of any size can be built, Figure 5.6, shows the various discrete points of the average quantization noise power at the output of the butterfly for different word-length combinations.

Input Constraints

Apart from the constraint on the total output performance, the fixed-point format of the input signals has to be taken into account while performing the optimization. Such input constraints arise in many practical scenarios in systems implemented using digital hardware or software interfacing with ADC[1] whose bit-widths are determined by other considerations such as standardization, analog circuit design etc. The presence of input quantization means two things: i) there is already some quantization noise that would contribute to the total output quantization noise, and ii) some of the signal formats are already fixed and they do not participate in the optimization.

Due to input quantization, the actual noise at the output added by the fixed-point operations within the sub-system is much less than the target total quantization noise power. So, the actual optimization has to be performed by setting the target quantization noise power to the difference between the total output quantization noise power and the contribution to the noise power due to quantized input. The input constraint contributes to a certain amount of quantization noise at the system output. The noise-power corresponding to this is the least magnitude of noise-power that can be achieved by using a fixed-point system. The corresponding cost of the system can be obtained by propagating the bit-widths across operators in the given system. Also, the input constraint narrows down the combinatorial search space.

Precedence Constraints

Each sub-system word-length optimization problem is solved for a specified noise budget independently of the other. When all the sub-systems are optimized, each of the sub-system signals are assigned a certain precision such that the noise-power criteria of sub-systems are satisfied. At this stage, even though the sub-systems are set to function at optimal word-lengths independently, they have to be made compatible with one another so that they work as a system. The compatibility issue arises of different word-lengths assigned to signals at the interface of each of these sub-systems. In other words, the noise contribution from each sub-system must be capable of generating the budgeted amount of noise-power into the system even in the presence of input biases from other sub-systems.

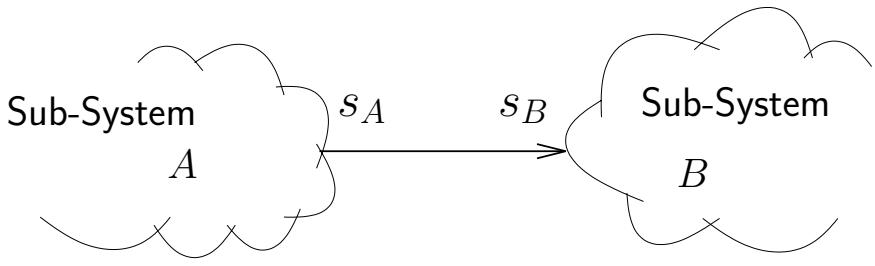


Figure 5.7: Mismatch between assigned word-length formats

[1] Analog to Digital Converters

Consider two sub-systems A and B with a common signal s as shown in Figure 5.7. While the signal s is an output from sub-system A, it is also an input to sub-system B. The sub-systems A and B are optimized independently and therefore the signal s participates in the optimization of both and can be assigned two different bit-widths in either cases s_A and s_B respectively. Depending upon the number of bits assigned to signal s, two scenarios occur when combining the various sub-systems

- Case 1: $s_A \geq s_B$
- Case 2: $s_A < s_B$

In the first case, the signal s has the same number of bits assigned to it by both the sub-systems or it is quantized with larger quantization step-size in sub-system B than in sub-system A. This incompatibility can be resolved by inserting a quantizer with suitable step-size. In the second case, the signal is already quantized with a larger step-size. Over-riding one of the decisions by assigning s_A or s_B bits affects the quantization noise and cost trade-off in both sub-systems.

The word-length mismatch problem can be overcome by optimizing sub-systems in a precedence order defined by the signal flow in the system. The sub-systems among which the quantization noise power has to be distributed can be carried out in the order of precedence defined by the flow of the signal in the system. Thereby, the input constraints of every sub-system is defined by its predecessor sub-systems in every iteration and the word-length of such signals is determined during the optimization of the sub-system from where the signal emanates. In example considered in Figure 5.7; since the signal s_A emanates from the sub-system A, the responsibility of finding optimal word-length for this signal is with the optimization procedure for sub-system A. Thereby eliminating the causation of the condition in where $s_A < s_B$.

5.3 Fixed-Point Refinement of a MIMO-OFDM Receiver

In this thesis, the example of a MIMO-OFDM receiver is chosen for word-length optimization. The high-level block diagram of the receiver system is shown in Figure 5.8. The parameters of this transmission scheme are summarized in Table 5.1. The rest of the conditions are assumed to be ideal.

Parameters	Value	Description
Receive Antennas	4	Number of antennas in the receiver
Transmit Antennas	4	Number of antennas in the transmitter
Number of Sub-carriers	64	Total number of sub-carriers in the OFDM symbol
Number of data carriers	48	Sub carriers used for actual data transmission
Cyclic Prefix	16	Cyclic repetition of time domain samples in order to avoid inter symbol interference

Table 5.1: Transmission scheme parameters of OFDM-MIMO

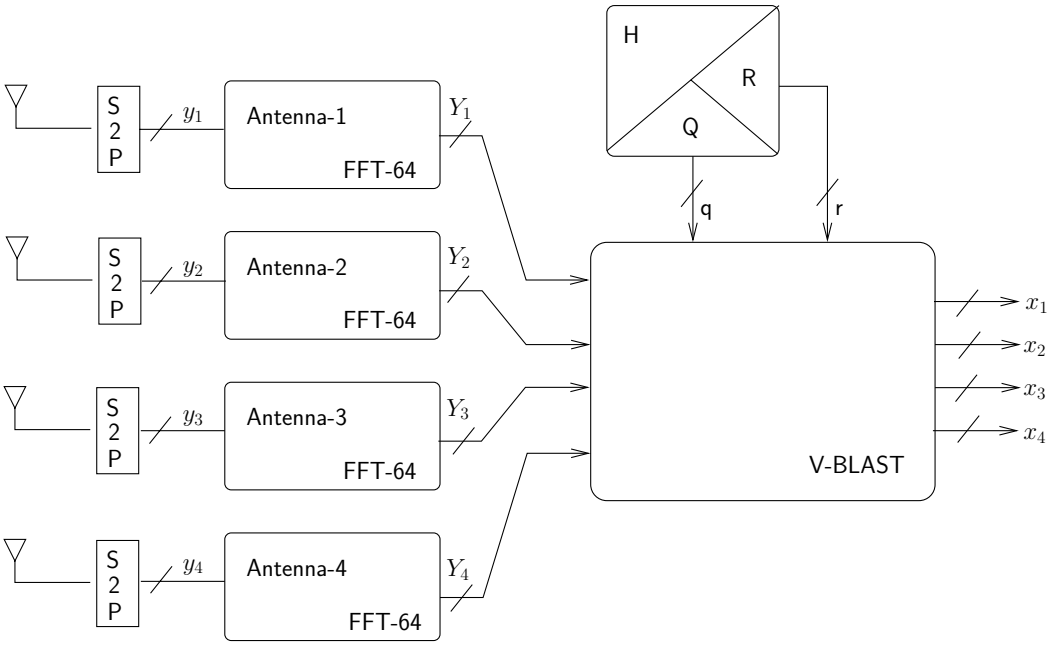


Figure 5.8: 4×4 MIMO-OFDM receiver system

This system consists of various functions involving arithmetic and decision operations. Functionally, there are three sub-systems: the FFT sub-system, the QR-decomposition sub-system and the V-BLAST sub-system. In the scheme of OFDM transmission, a 64 sub-carrier 4×4 MIMO, 16-QAM transmission scheme is considered. Four antennas are shown in the figure and the samples received are lined up in blocks of 64 by the serial to parallel converter: *S2P* blocks. The *S2P* block does not perform any computation and hence does not require any quantization noise considerations. The FFT sub-system computes 64 point fast Fourier transform. The QR-decomposition sub-system decomposes the 4×4 channel matrix into two matrices, Q and R which are 4×4 orthogonal matrix and a 4×4 upper triangular matrix. The V-BLAST sub-system performs the basic sphere decoding of the received signals to recover the transmitted 16-QAM symbols.

Each functional sub-system can be further decomposed into multiple levels of hierarchy. As a first step, these functional sub-systems are individually analyzed for suitability of applying the proposed *divide-and-conquer* technique. The hierarchical decomposition analysis and the optimization of the whole receiver system is presented towards the end of this chapter.

5.3.1 FFT

The radix-2 FFT is one of the widely employed signal processing algorithms. The signal flow graph of a N-point FFT and its hierarchical decomposition is shown in Figure 5.9. Here, the decimation in frequency [90] version of the FFT algorithm is

considered. Figure 5.10 shows the hierarchical decomposition tree of the FFT word-length optimization problem.

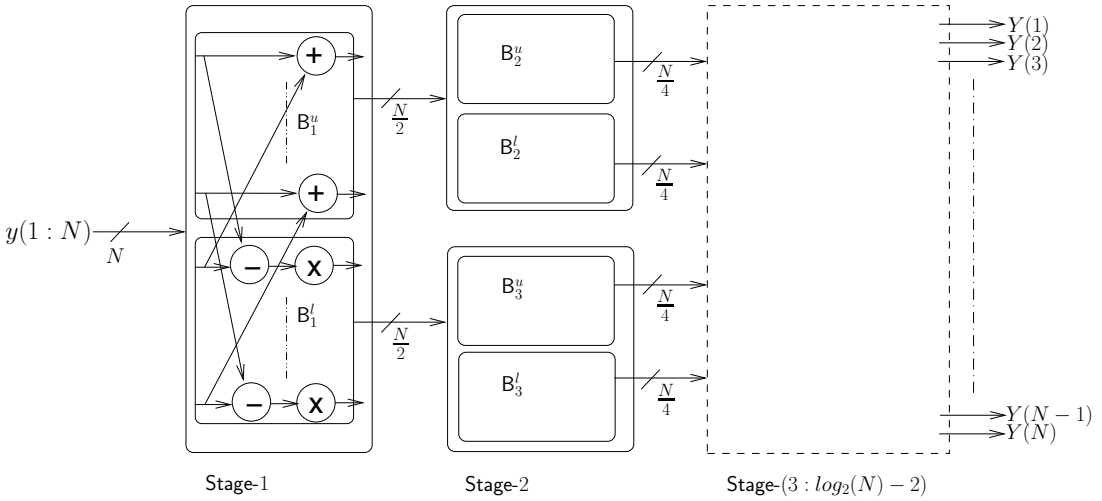


Figure 5.9: Hierarchical Decomposition of the FFT algorithm

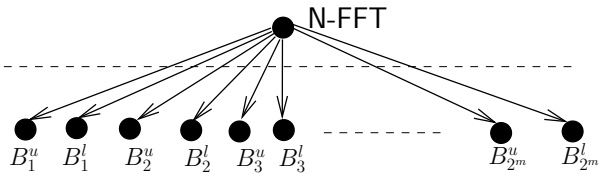


Figure 5.10: Hierarchical Decomposition tree of the FFT algorithm

An N-point, radix-2 FFT algorithm has $log_2(N)$ stages. Each stage can be thought of as a sub-system with N inputs and N outputs. Each of the stages are composed of $\frac{N}{2}$ butterfly operations. The operations performed in every successive stage are equivalent to performing the computations in the first stage of the FFT algorithm two times but with half the number of inputs. Thus, the r^{th} stage of an N-point FFT consists of 2^{r-1} first stages of an $\frac{N}{2^{r-1}}$-point FFT. It is clear from Figure 5.9 that the output from the first stage feeds into two independent $\frac{N}{2}$ FFTs and this repeats in successive stage.

The N-outputs of every stage are correlated with one another. In case of the first stage of an N-point FFT, the i^{th} butterfly outputs are captured as the i^{th} and the $\{i + \frac{N}{2}\}^{th}$ output of the first stage as More precisely, the first $\frac{N}{2}$ outputs are correspondingly correlated with the next $\frac{N}{2}$ outputs. Utilizing this homogeneity in the FFT structure, the first stage of the FFT operation is split into two sub-systems: *upper-half* consisting of all the addition operations and the *lower-half* consisting of the subtraction and complex multiplication of the butterfly operation. These are marked as blocks B_1^u and B_1^l in Figure 5.9. Each of these sub-system blocks feed into the first stage of $\frac{N}{2}$-point FFT. Following this scheme across all stages, the r^{th} stage of the

N-point FFT consists of 2^r sub-systems with 2^{r-1} sub-systems corresponding to the *upper-half* and an equal number of *lower-half* sub-systems.

In the hierarchical decomposition scheme, an N-point FFT consists of $2(N-1)$ number of sub-systems. This introduces $2(N-1)$ number of sub-system noise power variables in the optimization problem. Each of the sub-system noise power variables corresponds to the average noise power of all its outputs. A simple average is feasible as the sub-system outputs of both *upper-half* and *lower-half* sub-system types are uncorrelated. In contrast, the classical flat-approach requires each operation to be an optimization variable. Each butterfly consists of at least three operations corresponding to one complex addition, one complex subtraction and one complex multiplication. Therefore, the classical word-length optimization algorithm has $3\frac{N}{2}log_2(N)$ number of variables.

5.3.2 QR Decomposition

The CORDIC algorithm [83] is used to perform QR decomposition of a given matrix $\mathbf{H}$. In order to perform the decomposition, the CORDIC algorithm is used in two modes. In the vector mode, the CORDIC algorithm is used to measure the angle of a given vector. Input to the CORDIC algorithm in this mode is an ordered pair (X, Y) and the output is the corresponding polar co-ordinate representation (r, θ). In the rotation-mode, it is used to rotate a given vector by a specified angle. Input to the CORDIC algorithm in this mode is an ordered pair (X_i, Y_i) which is rotated by an angle θ to produce another (X_o, Y_o) in rectangular co-ordinate representation. The schematic of typical usage of the CORDIC algorithm for performing QR-decomposition is shown in Figure 5.11.

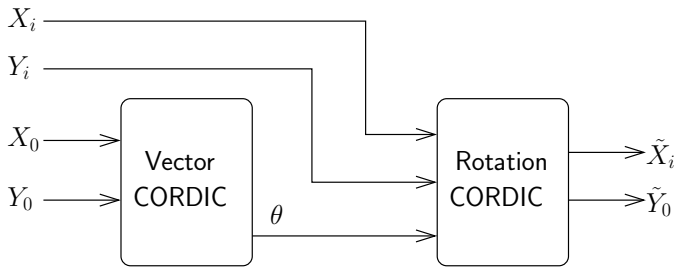


Figure 5.11: CORDIC for QR decomposition

The CORDIC algorithm involves repeated application of a computational transform so as to generate the desired coordinate transformations. Using a very high precision arithmetic system, the quality of the result improves with successive iterations. The computation and signal flow graph in both types of the CORDIC algorithm used for QR decomposition is essentially the same and is given as

$$x_i = x_{i-1} + d_i \cdot y_i tan(\phi_i); \tag{5.8}$$
$$y_i = y_{i-1} - d_i \cdot x_i tan(\phi_i); \tag{5.9}$$

where (x_i, y_i) corresponds to the output of the i^{th} iteration and ϕ_i corresponds to the angular step. The expression for $tan(\phi_i)$ is approximated by $\frac{1}{2^i}$. The signal-flow graph for one iteration of the CORDIC algorithm is as shown in Figure 5.12. This is repeated many number of times to achieve the required accuracy of angle of rotation of the vectors under consideration.

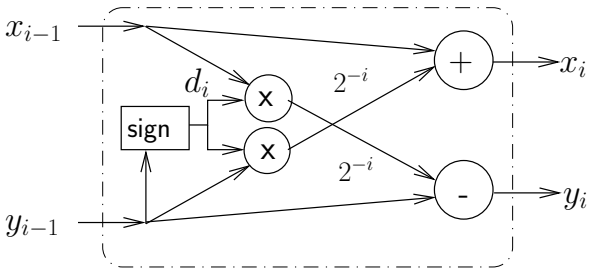


Figure 5.12: Computation data-flow in the vector mode CORDIC algorithm

In the vector-mode, the decision d_i is of unit magnitude whose sign is the same as Y_{i-1}. The sign of Y_i in successive iterations contributes to the angle of rotation θ. In the rotation-mode, the d_i is extracted from the sign information stored in the input θ.

The schematic in Figure 5.13 shows the signal flow graph for using the CORDIC algorithm to generate the **Q** and **R** matrices from the input matrix **H** such that

$$\mathbf{H} = \mathbf{Q} \cdot \mathbf{R} \tag{5.10}$$

The signal-flow-graph in Figure 5.13 generates the upper triangular **R** matrix. The **Q** matrix is obtained by successively multiplying the $\mathbf{Q}'$ matrix constructed after every vector mode operation. The SFG for the construction of **Q** is obtained by constructing the SFG for matrix multiplication and is not shown here. Also, the quantization noise introduced by the process of matrix multiplication is ignored in this experiment for the sake of simplicity.

The SFG for QR-decomposition of a 4×4 matrix essentially consists of three stages applied on the matrix **H** successively one after another. These three stages, which eventually generate the **R** matrix are shown in Figure 5.14. This also corresponds to the hierarchical decomposition of the SFG presented in Figure 5.13. The hierarchical decomposition tree of the word-length optimization problem is shown in Figure 5.15.

There are 32 iterations of the signal flow graph in Figure 5.12 used to perform every CORDIC rotation. A CORDIC vector operation is followed by as many CORDIC rotation operations as the number of columns in the matrix under consideration. One of the outputs of the CORDIC operator in the vector mode is ideally 0. Due to the

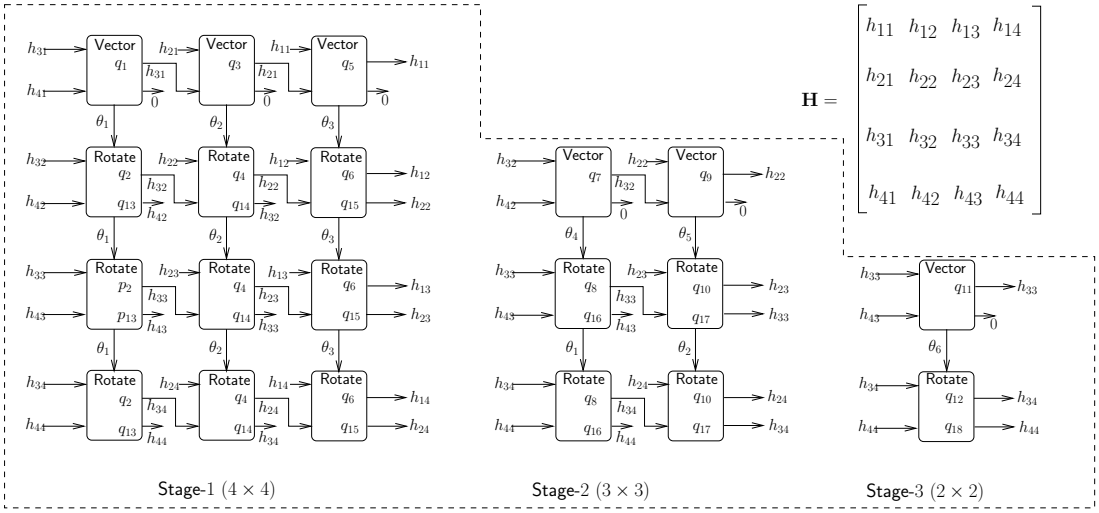


Figure 5.13: QR decomposition using CORDIC operations

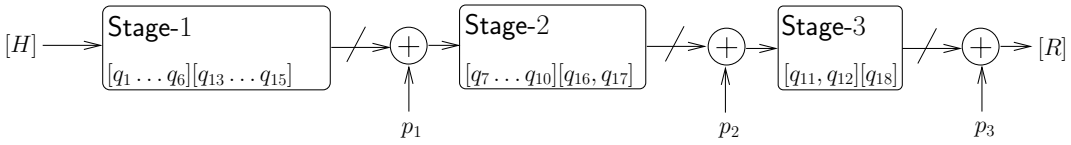


Figure 5.14: Hierarchical decomposition of the QR algorithm

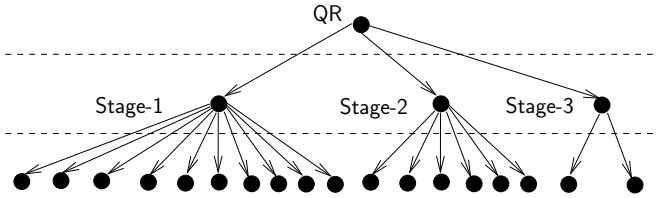


Figure 5.15: Hierarchical decomposition tree of the QR algorithm

errors due to CORDIC rotations, this value need not be zero. Since these output does not affect the computation, the non-zero values of these outputs can be disregarded. Mathematically, the non-zero output of the vector mode calculates the magnitude v and the angle θ of the vector represented by the two inputs as

$$v(x,y) = \sqrt{x^2+y^2} \tag{5.11}$$

$$\theta = tan^{-1}(\frac{y}{x}) \tag{5.12}$$

The rotation mode CORDIC operator consists of two outputs (c,d) representing the rotated input vector (x,y) by an angle θ. The mathematical expression is written

as

$$\begin{aligned} c(x,y,\theta) &= x \cdot cos(\theta) + y \cdot sin(\theta) \\ d(x,y,\theta) &= y \cdot cos(\theta) - x \cdot sin(\theta) \end{aligned} \tag{5.13}$$

The metric for evaluating the accuracy of QR decomposition is the difference between the original matrix $\mathbf{H}$ and $\hat{\mathbf{H}}$ obtained as $\hat{\mathbf{H}} = \mathbf{Q} \cdot \mathbf{R}$. The mean of the variance of each element of the matrix $\mathbf{\Delta H} = \mathbf{H} - \hat{\mathbf{H}}$ is used as the metric in this experiment. In this experiment, the entire CORDIC algorithm is used as operators instead of adders, multipliers and decision operators. These algorithms are essentially binary operators with two inputs and one output for the output v in the vector mode and outputs c and d in the rotation mode. Consider the computation of the correlation ρ_{cd} between the output of the rotation mode CORDIC operator.

$$\begin{aligned} \rho_{cd} &= \mathrm{E}(c(x,y,\theta) \cdot d(x,y,\theta)) - \mathrm{E}(c(x,y,\theta)) \cdot \mathrm{E}(d(x,y,\theta)) \\ &= \mathrm{E}((x \cdot y)\{cos^2(\theta) - sin^2(\theta)\} + \{y^2 - x^2\}(sin(\theta). \cdot cos(\theta)) \\ &\quad -\mathrm{E}(x) \cdot \mathrm{E}(y)\{\mathrm{E}(cos^2(\theta)) - \mathrm{E}(sin^2(\theta))\} \\ &\quad -\mathrm{E}(y)^2\mathrm{E}(sin(\theta))\mathrm{E}(cos(\theta)) + \mathrm{E}(x)^2\mathrm{E}(sin(\theta))\mathrm{E}(cos(\theta)) \\ &= \{\mathrm{E}(x \cdot y) - \mathrm{E}(x)\mathrm{E}(y)\} \cdot \{\mathrm{E}(cos^2(\theta)) - \mathrm{E}(sin^2(\theta))\} \\ &\quad +\{\mathrm{E}(y^2 - x^2) - \mathrm{E}(y)^2 + \mathrm{E}(x)^2\} \cdot \{\mathrm{E}(cos(\theta)) \cdot \mathrm{E}(sin(\theta))\} \end{aligned} \tag{5.14}$$

From the SFG, the input x and y to the first CORDIC operators are independent from each other. Therefore, the value of ρ_{cd} is always 0. Consequently, a noise power for each of the CORDIC operator outputs v, c and d is assigned. Since θ is shared by many rotation mode CORDIC operators, it is assumed that the rotation mode CORDIC operator have the same quantization noise behavior. Therefore, 9 noise powers in 3×3 configuration (one each for v, c and d occurring 3 times) for the first stage, 6 noise powers for the second and 3 noise powers for the third stage can be assigned. A flat approach therefore consists of 18 variables.

5.3.3 V-BLAST Decoding

The V-BLAST algorithm for a 4 antenna receiver consists of as many sub-systems. These sub-systems are separated from one another by a discriminator as shown in Figure 5.16. The computations of these sub-systems essentially consist of calculating

the inverse of the $\mathbf{H}$ to recover the transmitted signal $\tilde{\mathbf{x}}$ first by inverting $\mathbf{Q}$ as

$$\begin{aligned} \mathbf{y} &= \mathbf{H} \cdot \mathbf{x} \\ \acute{\mathbf{y}} &= \mathbf{Q}^T \cdot \mathbf{y} \\ &= \{\mathbf{Q}^T \cdot \mathbf{Q}\} \cdot \mathbf{y} \\ &= \mathbf{I} \cdot \mathbf{R} \cdot \mathbf{x} \\ &= \mathbf{R} \cdot \mathbf{x} \end{aligned} \tag{5.15}$$

The inversion of matrix $\mathbf{R}$ is obtained starting from the 4^{th} antenna to obtain $\tilde{x}_4$. This result is used to determine the value of $\tilde{x}_3$. Both $\tilde{x}_4$ and $\tilde{x}_3$ is used to determine the value of $\tilde{x}_2$. Further, the values of $\tilde{x}_4$, $\tilde{x}_3$ and $\tilde{x}_2$ is used to determine $\tilde{x}_1$ as shown in Figure 5.16.

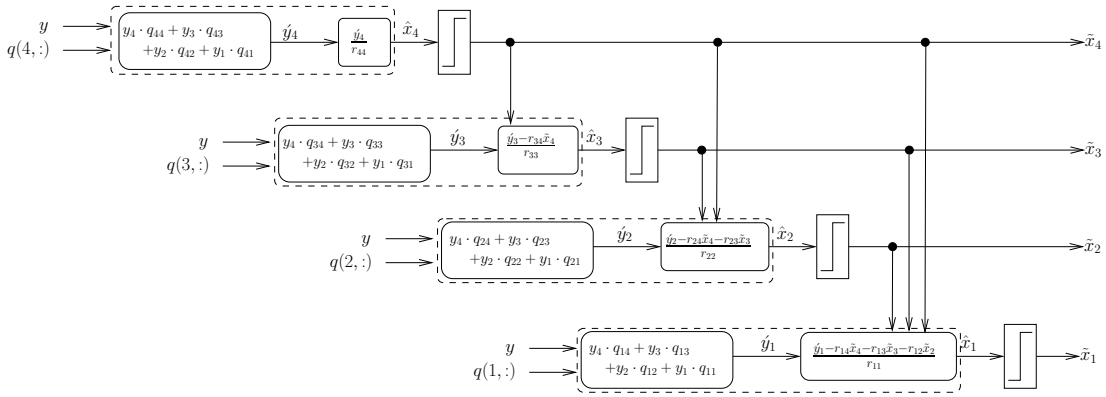


Figure 5.16: 4×4 V-BLAST algorithm

The decomposition tree of the V-BLAST algorithm is shown in Figure 5.17. It is clear from this tree diagram that the hierarchical word-length optimization problem has four optimization variables. In the flat approach, this algorithm consists of as many variables as the number of fixed-point operations which is 38 in number.

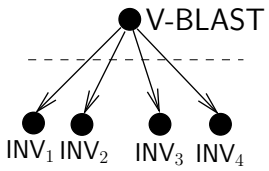


Figure 5.17: Hierarchical decomposition tree of the V-BLAST algorithm

5.3.4 Results: Flat vs. Hierarchical Optimizaiton approaches

In this section, the performance of the proposed hierarchical optimization algorithm in Section 5.2 and the classical *Min +1 bit* algorithm are compared in terms of the final

cost vs. accuracy trade-off achieved. As discussed in Section 5.2.3, both algorithms are iterative in nature and the exit condition requires that the quantization noise power criteria is met in both cases. The greedy nature of these algorithms and the non-convex nature of the word-length problem does not provide any control over the convergence of the solution. Therefore, very often these algorithms end up optimizing the fixed-point system with a better accuracy performance than what is required by the problem. Such solutions wastefully increase the cost while offering better accuracy when it is not required. These trade-off points can be thought of as solutions obtained for word-length optimization problems with a tighter accuracy constraint. In this thesis, such solutions are referred to as *over-optimized* solutions.

The FFT, QR decomposition and the V-BLAST algorithms are considered for illustrating the efficiency of the divide-and-conquer strategy. All experiments are carried out in the Matlab environment. In case of the FFT algorithm, it is possible to derive fully analytical expressions for estimation of quantization noise power using techniques described in Section 2.3.2. The QR algorithm uses CORDIC operations for performing rotations. The CORDIC operator is a fast and inexpensive way of either determining the magnitude of a given complex vector or to rotate the given vector by a given angle. It uses simple shifting and scaling operations to achieve this instead of using complex trigonometric functions. As a consequence, there is an error associated with the values obtained by the CORDIC algorithm with respect to trigonometric calculations. This effect together with degradation due to fixed-point quantization is studied exhaustively by simulation in this thesis. A detailed profiling of such operations is presented in Section 5.4.6. The input vector set with as many as 10^5 samples is considered for characterisation of CORDIC operations by simulation. These samples are sourced from a sample space which is distributed uniformly in the range $[-1, 1)$. The V-BLAST algorithm is implemented in Matlab and the *Hybrid* technique described in Section 4.6 is used to perform fixed-point performance evaluation. Here, 10^4 input samples from an unbiased 16-QAM source is considered for performing simulation using the *Hybrid* technique.

For all simulation based evaluations, sufficient number of points are used

It has to be noted here that the graph depicting the cost of the fixed-point system in Figure 5.18 is normalized with respect to the cost obtained by the flat approach. In Figure 5.19, the assigned target is set to the 100% mark. Due to the combinatorial nature of both hierarchical as well as flat approach, the actual performance obtained is 100% or better in which case, the performance graph raises to a point which is lesser than 100%. As shown in Figure 5.20, the number of iterations required for performing the global iterations reduces by nearly an order of magnitude in all the 5 cases considered. The optimization processes of each of the sub-systems in the MIMO-OFDM receiver is considered individually in the following sections.

FFT

In case of the FFT algorithm, the number of variables used for solving the global optimization problem is reduced by order of $O(log_2(N))$. The FFT algorithm with $4, 8$

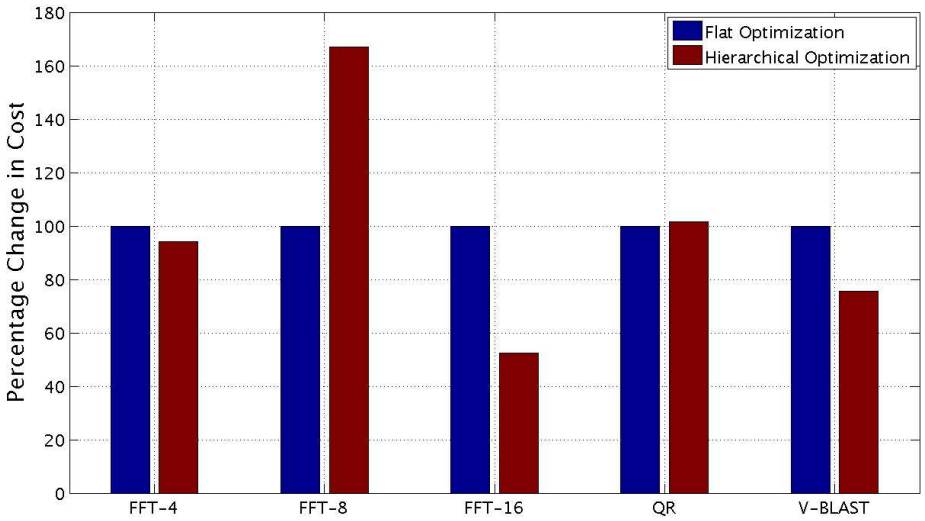


Figure 5.18: Comparison between cost of implementation obtained: flat vs. hierarchical optimization techniques

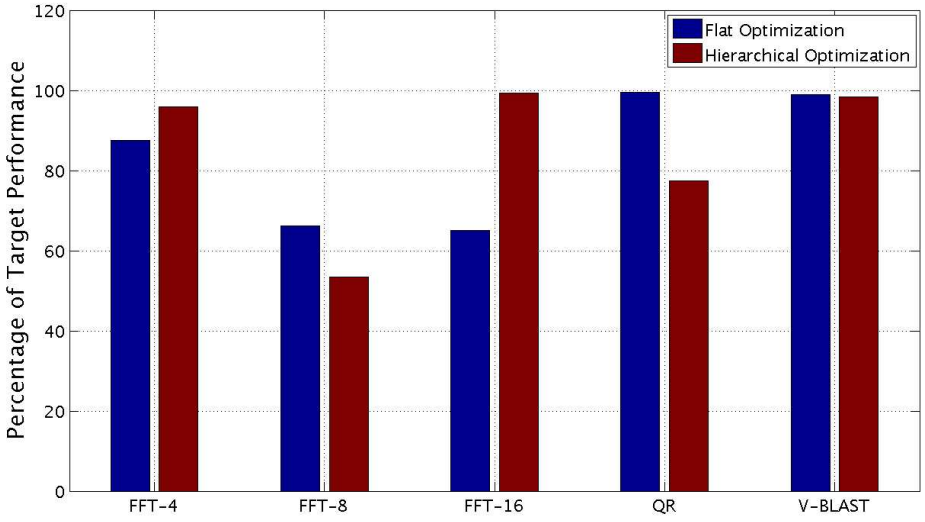


Figure 5.19: Comparison between target performance achieved: flat vs. hierarchical optimization techniques

and 16 points are considered. The total cost of implementation of the FFT algorithm and its total quantization noise performance obtained in all three cases showcase three different scenarios. In the FFT-4 case, the cost and the implementation are comparable

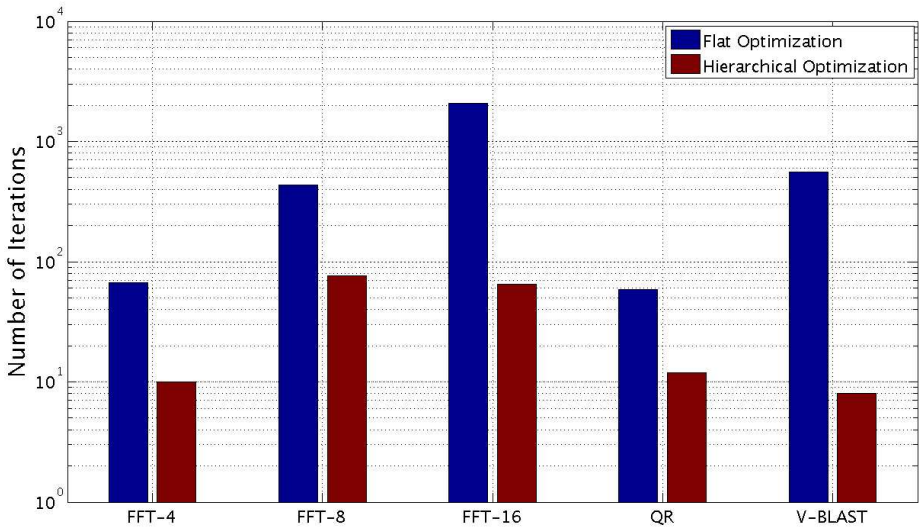


Figure 5.20: Comparison between the number of iterations: flat vs. hierarchical optimization techniques

with one another. In case of FFT-8, the cost in the hierarchical case is nearly 1.6 times higher than in the cost obtained by performing the classical word-length optimization. This is because of the over-optimization of performance by the hierarchical technique. In the third case of FFT-16, the quality of the results obtained in terms of cost vs. performance trade-off is diagonally opposite to that of case FFT-8. That is, the cost of implementation obtained by using the hierarchical optimization is as less as half the value obtained by the cost of fixed-point system optimized using the flat approach. In other words, the flat approach unnecessarily improves the over-optimizes the FFT algorithm in the third case. Optimizations of FFT algorithms with higher number of points are not considered in the interest of time required by the flat approach.

The reduction in number of iterations and the cost vs. performance trade-off achieved eventually is evident from the graphs shown in Figure 5.21 for the FFT-16 algorithm. The non-convexity of flat-optimization and hierarchical optimization is clearly visible by the shape of the curve traced by both curves. The starting points of both algorithms are different and it is dependent upon the maximum noise-power the sub-system power is allowed during the initialization phase. In this experiment, the noise-power step-size was set to $6dB$. Given the nature of the greedy heuristic adopted, it is not clear if the starting point or the actual choices available during iteration or the step-size is the reason for this difference in behavior. It could be a combination of all the three.

The actual execution times for word-length optimization on an Intel based Apple MacBook Pro [57] took approximately 8 minutes using the hierarchical approach while it took 33 minutes for the flat approach for the FFT-4 algorithm. The execution time

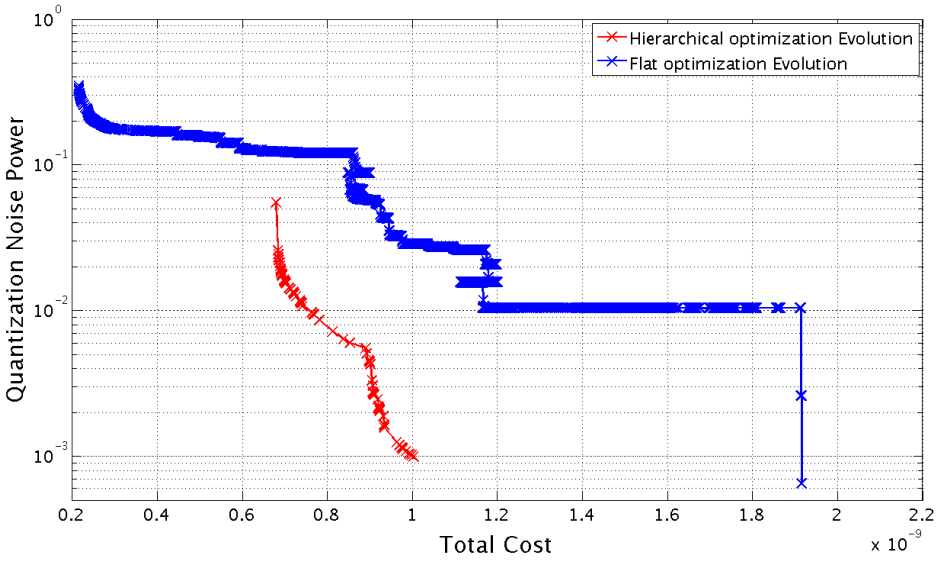


Figure 5.21: FFT-16: evolution of cost vs. performance trade-off

of FFT-8 and FFT-16 was 25 minutes and two hours respectively using the hierarchical approach. The flat approach took about six hours and upto about 14 hours respectively for FFT-8 and FFT-16 algorithms.

QR Decomposition

In case of the QR decomposition algorithm, the CORDIC operations used in both vector and rotation mode are used as basic operators. The number of variables in the flat approach is small for the *Min* +1 *bit* algorithm and is easily manageable by the flat approach. However, by adopting the divide-and-conquer strategy, this is reduced to just three variables. When the number of variables is less, the chances of occurrence of variables with similar trade-offs is greatly reduced. Under such circumstances, the performance of the *Min* +1 *bit* algorithm is generally better. This is reflected in the quality of solution obtained in both hierarchical and flat approach. Although the cost and performance are comparable, the total number of iterations required by the flat approach is greatly reduced.

The evolution of cost vs. accuracy trade-off in both hierarchical and flat approaches for the QR algorithm is shown in Figure 5.22. Because of the hierarchical approach, it is possible to take giant strides in every step with respect to the total quantization noise power to reach the desired quantization noise-power target. It is also because of these giant strides that the algorithm over-optimizes in its penultimate step to achieve a lower noise power level at the expense of higher cost. Thereby, making the solution obtained by the hierarchical process inferior to the one obtained by the flat approach.

The actual execution times for word-length optimization of the QR-decomposition

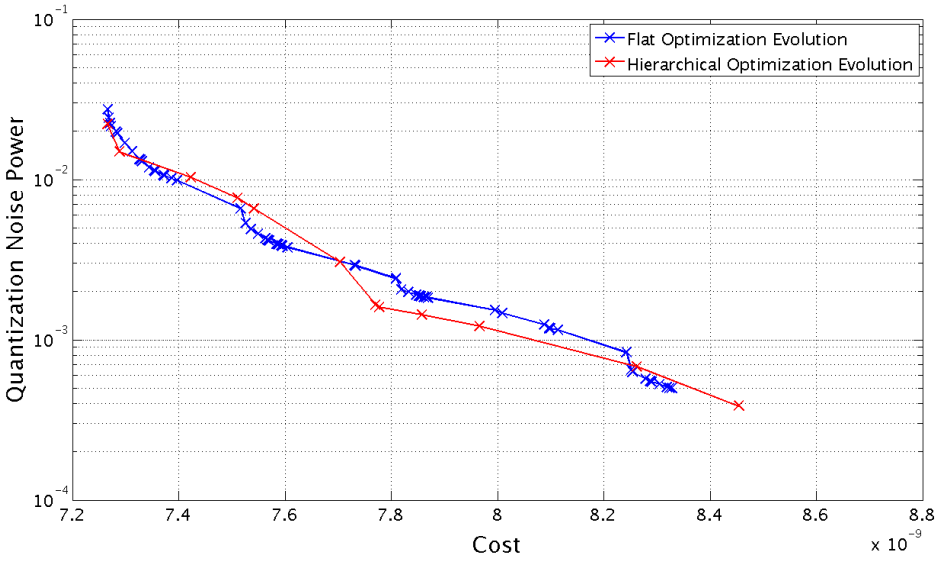


Figure 5.22: QR: evolution of cost vs. performance trade-off

algorithm on an Intel based Apple MacBook Pro [57] took about 18 minutes using the hierarchical approach while it took about 90 minutes for the flat approach.

V-BLAST

The V-BLAST algorithm consists of 38 additions and multiplications. Therefore, the classical approach for solving the word-length optimization problem consists as many optimization variables. The four sub-systems of the V-BLAST algorithm leaves just four variables. The cost achieved by using the hierarchical approach is about 80% of the cost obtained by flat approach. The performance of both fixed-point systems are comparable. This is also particularly true because, the target BER is a small quantity. Therefore, the difference seen is not huge. From the graph in Figure 5.20, there is nearly two orders of magnitude savings in the number of global iterations.

The evolution of cost vs. accuracy trade-off in both hierarchical and flat approach for the QR algorithm is shown in Figure 5.23. The bit error rate (BER) of the V-BLAST algorithm is a non-linear metric with respect to the quantization noise-power due to the presence of un-smooth operations. The steady increase in cost without any change in the BER is essentially because of this non-linearity. Indeed, this is the reason why there are steep drops as far as the flat optimization process is concerned. In the hierarchical technique, the noise added is substantial to make a change in the BER in every step. This is again due to the relatively large step size of $6dB$ chosen for the hierarchical optimization.

The hierarchical approachis effective in reducing the total number of global iterations. This reduces the time required for carrying out word-length optimization.

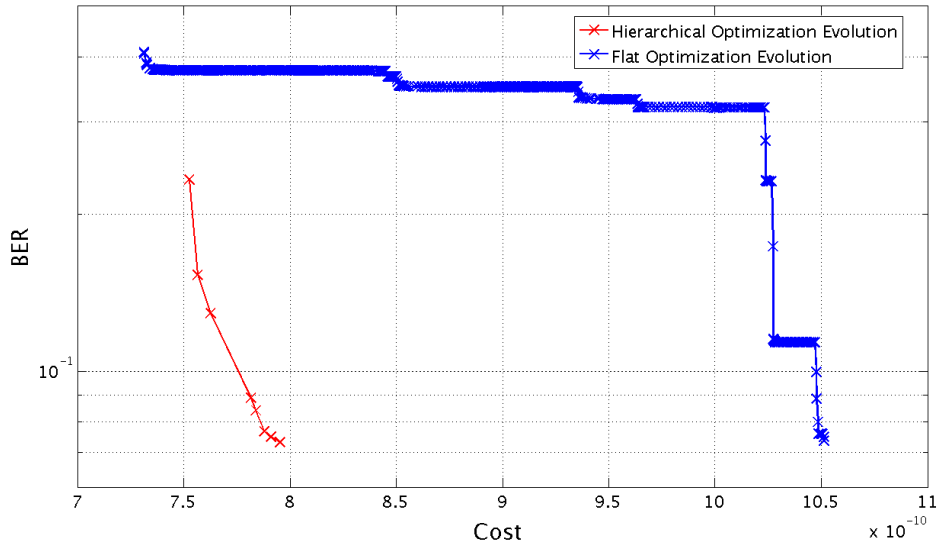


Figure 5.23: V-BLAST: evolution of cost vs. performance trade-off

Although the quality of the optimized solution obtained by the hierarchical and flat approaches are comparable, there are no good reasons to believe that one will always out perform the other. The quality of the solution is driven by the heuristics and influenced by the choices made during sub-system optimization. Since it is impossible to give any guarantee on the optimality of the sub-system optimization solution, nothing much can be said about the quality of the global optimization problem. In the next section, a convex optimization framework is proposed as an alternative heuristic to word-length optimization. Although the basic nature of the word-length optimization problem remains unchanged, an attempt to provide bounds on the quality of solution using the convex optimization framework is attempted.

The actual execution times for word-length optimization of the V-BLAST algorithm on an Intel based Apple MacBook Pro [57] took about 10 minutes using the hierarchical approach while it took about 14 hours for the flat approach to converge.

5.4 Convex Optimization Framework

From discussions in the previous section, it is clear that the global optimization problem is combinatorial in nature and there is no guarantee that the greedy heuristic will arrive at the optimal solution. In this section, an alternative technique for word-length optimization based on the principles of *Convex optimization theory* is proposed.

5.4.1 Previous Work

Analytical techniques such as [21], have attempted to cast the classical word-length optimization problem as a convex optimization problem. The cost minimization problem is written as

$$\text{minimize} \sum_{i=1}^{M} \omega_i b_i \quad \text{subject to} \quad \frac{1}{3} \sum \kappa_i 2^{-2b_i} \leq \kappa_{obj}, \tag{5.16}$$

where b_i is the number of bits corresponding to the i^{th} signal, κ_i is the path gain from the i^{th} signal to the output and ω_i is the cost of using b_i bits for the i^{th} signal. The path gain is calculated analytically by techniques discussed in Chapter 2.

The Objective Function

The authors in [21], relax the integer constraint on the number of bits b_i on any i^{th} signal and make them real valued. Due to this relaxation, the cost estimation function (objective function) of the minimization problem in Equation 5.16 is convex when the weights are kept constant throughout the optimization process. It has to be noted that the actual cost function is not this simple always. In a more practical scenario, it is not possible to express the cost function by scalar multiplication of weights as expressed in the minimization problem in Equation 5.16. Even if this were to be possible, the cost weights (ω_i) need not be the same for all possible word-length assignments.

To illustrate this, consider the cost of a binary operator such as a two input adder. One way of implementing such a fixed-point adder is to perform addition by using a full adder circuit with as many bits as the maximum bits assigned to either inputs and then discard the resulting bits either by truncation or rounding. Suppose the average energy dissipated as a function of number of bits is used as the cost metric. The total cost C_a of using such an adder circuit depends on the number bits assigned to each of the inputs signals and it is written as

$$C_a \propto \max(b_1, b_2), \tag{5.17}$$

where $\propto$ represents proportionality and b_1, b_2 are the word-lengths assigned to the inputs of the adder. Although the objective function in Equation 5.16 is convex, it does not represent the actual cost function.

The Constraint Function

There can be more than one quantization noise source along a given path to the output. The noise contribution by each fixed-point quantization noise source is not just a function of that particular fixed-point word-length. Consider the data path shown in Figure 5.24, the quantization noise power added by two quantizers is shown in the equivalent additive PQN model.

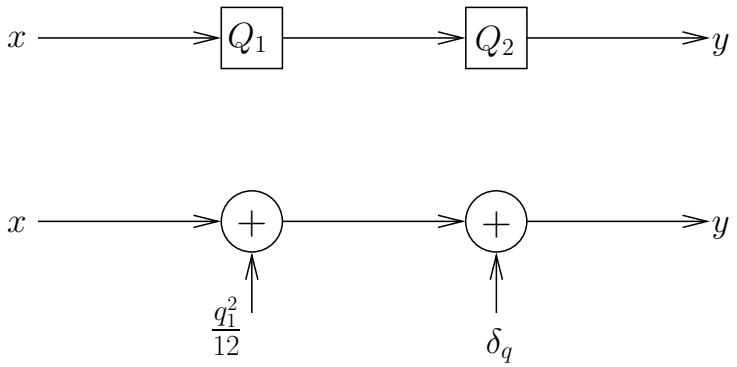


Figure 5.24: Quantization noise sources along a data-path

As given in Table 2.1, the amount of quantization noise added by the second quantization in the quantizer Q_2: the value of δ_q does not depend on the number of bits assigned at the output of Q_2 alone. It is indeed a function of both quantization step sizes q_1, q_2 and is given as

$$\delta_q(q1, q2) = \begin{cases} \frac{q_2^2}{12} - \frac{q_1^2}{12} & q_2 > q_1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.18}$$

Only when $q_1 << q_2$, the value of δ_q approaches the value of $\frac{q_2^2}{12}$.

Clearly, it is only under such conditions that the constraint function in the minimization problem formulation in Equation 5.16 can be approximated to be convex by ignoring the noise contribution by the quantizer Q_1. In practice, this can happen in scenarios where the difference between two quantizers is very large such as a data-path where a multiplier is followed by an adder. In cases where the difference between the two step-size is as small as 1 bit, ignoring the quantization noise contribution by the first quantizer introduces an error of nearly 25% in the estimation.

In summary, the problem formulation as given in Equation 5.16 would only work well if the quantization step sizes between successive quantization are relatively large and the cost function was as simple as a bit-count. It fails to capture all the nuances of the quantization dynamics with respect to both the cost objective function and the performance constraint function.

5.4.2 The Noise Budgeting Problem

In the hierarchical approach described in section 5.1, the word-length optimization problem attempts to budget the total noise-power at the system output to noise-power contribution from every sub-systems. The noise-budgeting view-point can be scaled down to the operator level. The cost minimization problem can then be written as

$$\min(C(\mathbf{q})) \quad \text{subject to} \quad \lambda(\mathbf{q}) < \lambda_{obj}, \tag{5.19}$$

where $\mathbf{q} = [q_1, q_2 \dots q_N]$ corresponds to the noise-power injected into the system by N different operators used to implement the fixed-point design. The use of symbol q_i instead of p_i is to differentiate it from the fact that this problem deals with the quantization noise injected at each operator level and not at the sub-system output. In order to solve the noise budgeting problem, it is necessary to study the trade-off behavior cost and noise-power of all types of operators in the sub-system for various choices of fixed-point configurations.

It is possible to profile both the cost and quantization noise contribution as a function of given fixed-point word-lengths assigned to their inputs and outputs. All basic operators used in the design and implementation practices of signal processing applications are simple with few inputs and outputs. Therefore, it is relatively easy to conduct an exhaustive profiling by considering all possible combinations of fixed-point word-length on each individual operator. These operators can be something as common as an adder or a multiplier or it could be any special operator such as the CORDIC operator.

By conducting an exhaustive profiling of the entire fixed-point combinatorial space for every type of operations used in the implementation of fixed-point system, it is possible to arrive at a function which relates the cost of using a fixed-point operator as a function of its quantization noise power. This function can be used for making the trade-off between the quantization noise contribution and the cost of the given operator.

Operator Level Trade-Off

Consider a fixed-point operation of type d with N inputs and M outputs. In order to exhaustively profile the cost vs. accuracy trade-off of this operator, a word-length vector $\mathbf{w}_d$ of size $N + M$ is considered. Let W be the number of different word-lengths that can be assigned to each of the signals of this operation. The complexity of exhaustively searching through the entire search space of one such operator is of the order $O((N + M)^W)$. For example, consider the case of a binary adder which can be assigned anywhere between 2 bits and 16 bits to its fractional part. Then, $N = 2$, $M = 1$ and $W = 14$ which leads to 3^{14} unique permutations of the word-length vector $\mathbf{w}_d$. It is possible to evaluate the cost and accuracy of every operator type d because: i) the vector $\mathbf{w}_d$ word-length vector is not expected to be large, ii) functions evaluating cost and noise-power are relatively simple and iii) the number of different types of operators are few in number.

Figure 5.25 shows all the points considered during an exhaustive search of a binary adder. Three signals: two inputs and one output of the binary adder were assigned all permutations of bit-widths ranging from 1 bit to 24 bits. The performance of every fixed-point operation can be obtained either by simulation or by analytical techniques. Energy dissipation cost of the fixed-point operator is obtained by looking up from a

hardware library developed during the thesis work [54]. The total energy dissipation cost of the given implementation E is obtained as

$$\mathrm{E} = \sum_{i=1}^{N} \mathrm{E}_{op}^{i} \cdot n_{op}^{i}, \tag{5.20}$$

where N is the number of different types of operators qualified by their fixed-point word-lengths and E_{op}^{i} is the corresponding energy dissipation obtained by looking up the library and n_{op}^{i} is the number of the i^{th} type of fixed-point operator.

To build the library, the energy dissipation of binary adders and multipliers with various fixed-point configurations is build on the targeting the ASIC platform of $130nm$. The energy consumption estimates are obtained by using *Prime Time* from Synopsys [117] and the estimates obtained are in Joules. To make the measured energy dissipation data agnostic, a random input test vector set which is uniformly distributed and which spans the entire range of the assigned binary fixed-point range is used. Therefore, the energy dissipation values correspond to the average energy dissipated.

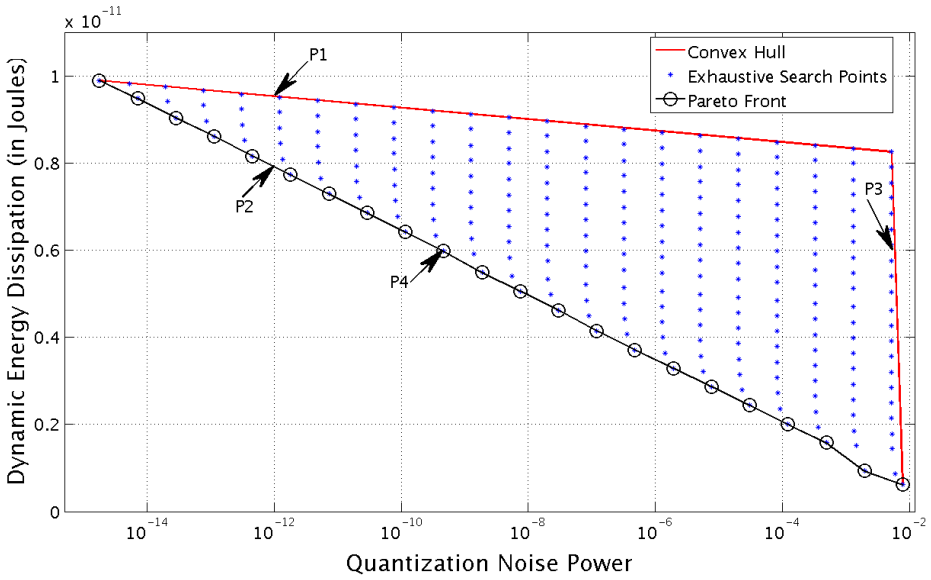


Figure 5.25: Adder: cost vs. accuracy trade-off in the semilog scale

In the above figure, the y-axis shows the energy dissipation cost and the x-axis shows the quantization noise introduced by the operator for each of the points considered. The y-axis is plotted on a linear scale for measuring the energy dissipation cost and the x-axis is plotted on the logarithmic scale to plot quantization noise.

The *convex-hull* is a convex polygon encompassing all the points considered during the exhaustive search. This polygon is drawn around the points plotted on a graph with linear scale on either axes. The logarithmic scale for x-axis is chosen for the purpose of illustration and for clear visibility of all points. In Figure 5.26, the convexity of

the polygon is clearly visible when the linear scale for representing noise-power on the x-axis is chosen.

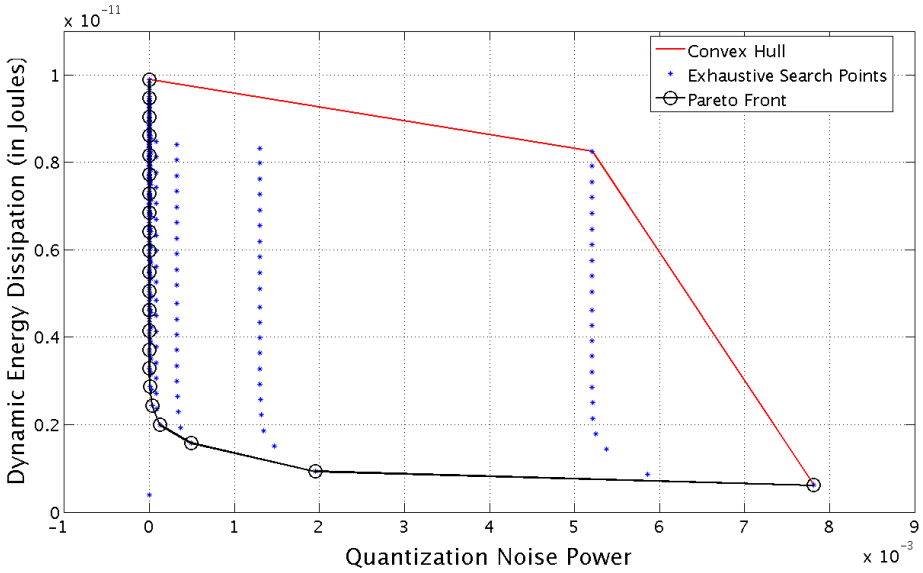


Figure 5.26: Adder: cost vs. accuracy trade-off in the linear scale

Identifying *Pareto front*

The choice of points that form the Pareto front should be chosen such that they do not make sub-optimal choices. For example, consider points P_1 and P_2 that contribute equal quantization noise-power but with different costs as shown in Figure 5.25. Although both points lie on the convex hull, it is clear that the cost of P_1 is higher than P_2 and therefore choosing P_1 is sub-optimal. Similarly, consider points P_3 and P_4 that also lie on the convex hull and have the same cost. The point P_3 cannot be an optimal choice because it contributes higher quantization noise power than P_4 for the same cost. Hence, it is clear that those points that lie along the convex-hull closer to the origin are the ones representing the minimum cost for a given quantization noise-power. The line joining all such points mark the Pareto-front of the operator under consideration.

In Figures 5.25 and 5.26, the *Pareto-front* for the adder used in this thesis is shown. It has to be noted that there are several more combinations of the word-lengths that can be assigned to the operator. Those combinations that do not contribute to the total quantization noise of the fixed-point system are not shown in the plot for the sake of clarity. For any combination of input bit-widths, the output word-length such that the total quantization noise is 0 can be obtained by simply propagating the assigned bits across the operators. Moreover, increasing cost which does not improve accuracy is not useful. Hence, the points corresponding to 0 noise-power are neither considered nor marked on the graph.

5.4.3 Relaxation for convexity

The *Pareto-front* boundary marked in Figure 5.25 is a continuous line $\Phi(q)$ as a function of quantization noise power q obtained by connecting successive points along the convex-hull with straight lines. These lines are also the lines of the convex polygon. Therefore, the function $\Phi(q)$ traces a piece-wise linear curve. Using some of the fixed-point configurations of the adder, only some points on the points on this line can be realized. On the contrary, if all the points on the continuous line were to be realizable, it would provide the optimal cost vs. accuracy trade-off. Therefore, the idea here is to relax the constraint on the integer nature of the bits such that all points on the *Pareto-front* become valid. Then, the word-length optimization problem can be solved by using standard convex optimization techniques to realise an optimal working point on the continuous *Pareto-front*. Eventually, the discrete points realizable by using integer word-lengths nearest to the *Pareto-front* under integer word-length constraint can be chosen for actual realization of a system with integer word-lengths.

If q is the quantization noise contributed by the operation under consideration, let κ be the corresponding minimum cost of the operator. The value of κ can be obtained by looking up the *Pareto-curve* and is given as

$$\kappa = \Phi(q) \tag{5.21}$$

The cost function in the minimization problem in Equation 5.23 is essentially the sum of individual operator costs. This assumption is still not representative of different practical cost function scenarios such as the impact of binding and scheduling and other resource sharing overheads. However, it is more realistic than just counting the number of bits. This cost metric is applicable to the energy dissipation cost of parallel hardware designs considered in this thesis. So, consider a system consisting of N operators. The total system cost can be written as

$$\begin{aligned} C_i(\mathbf{q}) &= \sum_{i=1}^{N} \kappa_i, \\ C_i(\mathbf{q}) &= \sum_{i=1}^{N} \Phi_i(q_i), \end{aligned} \tag{5.22}$$

where $\Phi_i(.)$ is the function capturing the *Pareto-front* of the of the i^{th} operator.

5.4.4 Convexity of the noise budgeting problem

In order to use convex optimization techniques to solve the minimization problem in Equation 5.19, it is important to check if the problem considered is indeed a convex optimization problem. In the light of the previous section, minimization problem of i^{th} sub-system with N quantization noise sources can now be written as

$$\text{minimize} \left(\sum_{j=1}^{N} \Phi_i(q_i) \right) \text{ subject to } \lambda(\mathbf{q}) \leq \lambda_{obj}, \tag{5.23}$$

where $\lambda(\mathbf{q})$ is the total quantization noise at the output of the system, obtained from Equation 5.25 and λ_{obj} is the target accuracy objective. The minimization problem is a convex optimization problem if both objective and constraint functions are convex.

From first principles, a function $f(x)$ is convex if the domain $dom(f)$ is a convex set and for all $m, n \in dom(f)$ and $\eta \in (0, 1)$ the following relation holds [12]

$$\eta \cdot f(m) + (1 - \eta) \cdot f(n) \geq f(\eta \cdot m + (1 - \eta) \cdot n) \tag{5.24}$$

Performance Evaluation Function

Using the linear noise propagation model described in Chapter 2, the total noise power at the output of the system is the sum of all operator noise powers scaled by their respective path gains. The total noise at the output of the system as a function of the noise power vector $\mathbf{q}$ is given as

$$\lambda(\mathbf{q}) = \underbrace{\sum_{i=1}^{N} \beta_i \sigma_i^2}_{\sigma^2} + \underbrace{\left(\sum_{i=1}^{N} \alpha_i \mu_i \right)^2}_{\mu^2}, \tag{5.25}$$

where σ_i and μ_i are the variance and the mean of the quantization noise power q_i generated by the i^{th} operator. The function $\lambda(\mathbf{q})$ is a function of all the noise-power generated within the system. Let q_i^{σ} be the standard deviation of the quantization errors and $q_i^{\mu} = \sqrt{\mu_i}$ of the i^{th} operator. The noise source q_i consists of contribution from the respective variance and mean components as

$$\begin{aligned} q_i &= q_i^{\sigma} + q_i^{\mu} \\ &= \sigma_i^2 + \mu_i^2 \end{aligned} \tag{5.26}$$

The total quantization noise power function $\lambda(\mathbf{q})$ at the output of the system can also be split into two and expressed as the sum of two functions $\lambda^{\sigma}(\mathbf{q}^{\sigma})$ and $\lambda^{\mu}(\mathbf{q}^{\mu})$, where $\mathbf{q}^{\sigma} = [q_1^{\sigma}, q_2^{\sigma} \ldots q_N^{\sigma}]$ is a vector of the noise source components corresponding to contribution by noise variance component and $\mathbf{q}^{\mu} = [q_1^{\mu}, q_2^{\mu} \ldots q_N^{\mu}]$ is a vector of noise contribution by the mean component. The total noise-power as a function of the mean and variance components of the operator noise-sources is written as

$$\begin{aligned}\lambda(\mathbf{q}) &= \lambda^{\sigma}(\mathbf{q}^{\sigma}) + \lambda^{\mu}(\mathbf{q}^{\mu}) \\ &= \sum_{i=1}^{N} \beta_i q_i^{\sigma} + \left(\sum_{i=1}^{N} \alpha_i \sqrt{q_i^{\mu}} \right)^2 \\ &= \sum_{i=1}^{N} \beta_i q_i^{\sigma} + \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k \sqrt{q_i^{\mu} q_i^{\mu}}\end{aligned} \tag{5.27}$$

If the function $\lambda(\mathbf{q})$ is convex, it has to satisfy the condition for convexity in Equation 5.24. The expression corresponding to the right-hand-side (RHS) of the convexity condition is written as

$$\begin{aligned}\lambda(\eta\mathbf{m} + (1-\eta)\mathbf{n}) &= \\ &= \sum_{i=1}^{N} \beta_i (\eta \cdot m_i^{\sigma} + (1-\eta) \cdot n_i^{\sigma}) + \\ &\quad \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k \sqrt{(\eta \cdot m_i^{\mu} + (1-\eta) \cdot n_i^{\mu})(\eta \cdot m_k^{\mu} + (1-\eta) \cdot n_k^{\mu})},\end{aligned} \tag{5.28}$$

where $\mathbf{m} = [m_1, m_2, \ldots, m_N]$ and $\mathbf{n} = [n_1, n_2, \ldots, n_N]$ are two combinations of the quantization noise power source vector $\mathbf{q}$ such that it is an optimal solution to the noise budgeting problem for two corresponding accuracy constraints λ_{obj}^{m} and λ_{obj}^{n} respectively. The gain $\beta_i = \mathrm{E}[h_i^2]$ is the expectation of the impulse response of the path function from the i^{th} source to the output. Therefore, β_i cannot be negative. The *square-root* operation for the mean part makes the noise estimation function non-convex. Therefore, in general the constraint function is not convex. In the case of convergent rounding, mean of the quantization noise is exactly zero. It is very close to zero even if it is the case of simple rounding. That is, the noise contribution due to mean is either zero or is negligibly small in the rounding. In such a scenario, continuing with evaluation of the expression in Equation 5.28 it can be written as

$$\begin{aligned}\lambda(\eta\mathbf{m} + (1-\eta)\mathbf{n}) &= \\ &= \eta \left(\sum_{i=1}^{N} \beta_i m_i^{\sigma} + \right) + (1-\eta) \left(\sum_{i=1}^{N} \beta_i n_i^{\sigma} \right) \\ &= \eta \cdot \lambda(\mathbf{m}) + (1-\eta) \cdot \lambda(\mathbf{n})\end{aligned} \tag{5.29}$$

Therefore, it can be concluded that the accuracy evaluation function in the rounding

case is not only convex but is also affine. Considering the result in Equation 5.29, the technique described henceforth is strictly applicable only to quantization carried out in the convergent rounding mode. However, as the magnitude of rounding is very small in the simple rounding case, this result approximately holds true even for simple rounding mode. In the truncation mode, the noise power contribution by the mean component is comparable to the variance component. Therefore, this proposal is not applicable for truncation mode quantization.

Cost Function

In this section, the cost function defined in Equation 5.23 will be proved to be a convex function. The function $\Phi(x)$ is obtained by an exhaustive profiling of the operator and considering the Pareto-front obtained by constructing a convex polygon around the points thus obtained on the cost vs. accuracy axes. Therefore, it is convex by definition. The convexity of the function $\kappa_i = \Phi_i(q_i)$ for every operator i implies that

$$\Phi_i(\eta \cdot q_i^m + (1-\eta) \cdot q_i^n) \quad \leq \quad \eta \cdot \Phi_i(q_i^m) + (1-\eta) \cdot \Phi_i(q_i^n) \tag{5.30}$$

Now, consider evaluating the right-hand-side of Equation 5.24 with respect to the cost estimation function $C(\mathbf{q})$.

$$\begin{aligned} C(\eta\mathbf{m} + (1-\eta)\mathbf{n}) & \\ &= \sum_{i=1}^{N} \Phi_i(\eta q_i^m + (1-\eta) q_i^n) \\ &\leq \sum_{i=1}^{N} \eta \Phi_i(q_i^m) + (1-\eta)\Phi_i(q_i^n) \\ &= \eta \sum_{i=1}^{N} \Phi_i(q_i^m) + (1-\eta) \sum_{i=1}^{N} \Phi_i(q_i^n) \\ &= \eta C(\mathbf{m}) + (1-\eta) C(\mathbf{n}) \end{aligned} \tag{5.31}$$

Therefore, the cost function at the sub-system level is convex.

5.4.5 Near-Optimal Word-length Optimization Algorithm

Using the relaxation technique discussed in the previous section, the problem of cost minimization subject to accuracy constraint of a fixed-point system can be casted as a convex optimization problem. In this section, a word-length optimization algorithm that captures the various steps of the convex relaxation process in order to use standard convex optimization solvers and apply the result obtained on the word-length optimization problem is presented.

Consider any i^{th} sub-system within the given system. The algorithm in Procedure 5.2 describes different steps to use the convex optimization framework for solving the sub-system word-length optimization problem.

Procedure 5.2 : Near-Optimal Word-length Optimization

1: $S_i(V,E)$ = GetSubSystemGraph()
2: N_t = GetOperatorTypes($S_i(V,E)$)
3: **for all** n_i Operator types $n_i \in N_t = [n_1, n_2, \ldots, n_t]$ **do**
4: db_j = ExtractOperatorPoints(n_i, Wd_{Min}, Wd_{Max})
5: Φ_j = GetConvexParetoFront(db_j)
6: **end for**
7: C_i = GetCostExpression($S(V,E)$)
8: λ_i = GetNoisePowerExpression($S(V,E)$)
9: $\mathbb{P}$ = ConstructMinimizationProblem (db,λ_{obj},C_i, λ_i)
10: $\bar{q}_{opt}$ = Solve($\mathbb{P}$)
11: $\bar{Wd}_{opt}$ = GetFinitePrecisionWordlengths($\bar{Wd}_{in}$, $\bar{q}_{opt}$, $\bar{q}_{opt}$, P_{obj},db)

The first step in solving the word-length optimization problem is to obtain the data flow graph $S(V,E)$ consisting of V nodes and E edges by calling the function **GetSubSystemGraph()**. The graph S is a directed graph with one operator at each node position, the edges connect various operators and point in the direction of the data-flow in the algorithm. The various operator types are enumerated for studying the cost and accuracy trade-off behavior. The function **ExtractOperatorPoints()** conducts an exhaustive search of the operator and returns all the feasible operating points which will be stored in the operator database *db*. As described in section 5.4.3, the convex *Pareto-front* Φ_i of every type of operator is deduced from the trade-off points by the function **GetConvexParetoFront()**.

Analytical expressions for the sub-system cost and noise power is determined by the techniques presented in Chapter 2 and analytical expressions for the same are obtained by calls to functions **GetCostExpression()**, **GetNoisePowerExpression()** respectively. The word-length optimization problem is expressed using the standard optimization modeling language such as *"CVX"* and solved by employing any standard solvers such as *"SeDumi"* or *"SDPT3"*. This step is performed in the function **ConstructMinimizationProblem()** to obtain the minimization problem $\mathbb{P}$. The procedure **Solve()** essentially calls the solver to solve the minimization problem $\mathbb{P}$. Solving the convex optimization problem, the minimum cost of implementation is obtained such that the performance constraint is satisfied. The values of noise powers of each operator which gives the minimum cost is got in the vector $\bar{q}_{opt}$. The individual operator cost can be deduced by performing an inverse of the operator level pareto curve (i.e. $\Phi_i(q_i)$ for the i^{th} operator). In the final step, the procedure **GetFinitePrecisionWordlengths()** realises the budgeted optimal noise-power using fixed-point operators.

Realising Noise-power Using Fixed-Point Operators

Due to the discrete nature of the fixed-point word-length assignment, as discussed in section 5.4.2, the noise-power corresponding to the optimal trade-off point need not always be realizable using fixed-point operators. Therefore, a feasible point (W_d) from the operator trade-off database in the locality of the optimal point such that all the input constraints are satisfied need to be considered.

Feasible trade-off points for using the fixed-point operator in various input and output bit-precision configurations is already pre-computed and stored in the respective operator databases. When the optimal noise-power suggested by the convex optimization solver coincides with one of the feasible points in the operator data-base, the operator precision at its output is precisely found. In such circumstances, Wd_i is set to the fixed-point word-length corresponding to the exact match. If that is not the case, the nearest feasible point is chosen such that it does not affect the sub-system output accuracy constraint greatly.

The *Pareto-optimal curve* of an adder is plotted in Figure 5.27. This figure also shows the actual trade-off points corresponding to three instance of using the adder with different input quantization constraints. The semilogarithmic scale is chosen to clearly identify the various trade-off points.

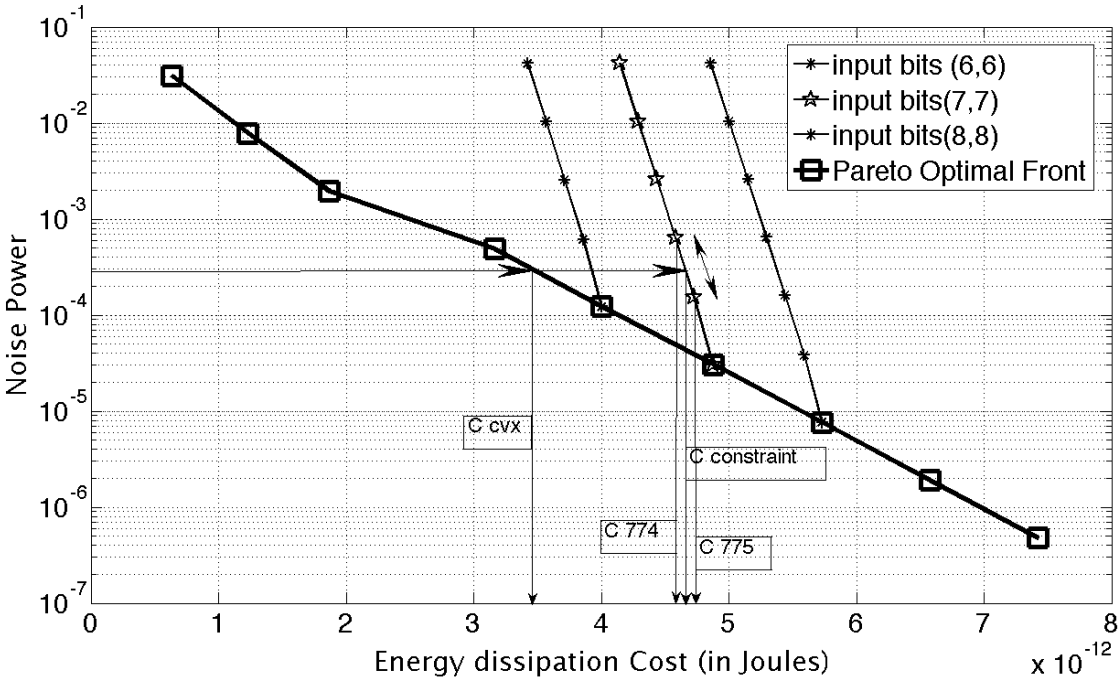


Figure 5.27: Finding Feasible operating points

Consider a particular case of an adder where the quantization noise assigned by the convex optimization process is $3 \cdot 10^{-4}$. The minimum cost as attained by the convex

optimization algorithm is obtained by calculating the inverse of the convex-pareto curve as $C_{cvx} = \Phi(3 \cdot 10^{-4})$. Clearly, this is not a feasible point. Further, suppose the input constraints to this operator is $(7, 7)$ (i.e. both inputs are quantized by 7 bits). With the input constraints applied, it is clear that the trade-off point must lie on the vertical line corresponding to the input constraints $(7, 7)$. After translation, the cost of the operator is $C_{constraint}$. The point is still non-feasible as fractional word-length assignment is not possible. Therefore, the nearest feasible point satisfying the noise-power constraint is chosen. Two points $(7, 7, 4)$ and $(7, 7, 5)$ exist in the vicinity corresponding to this quantization noise-power.

Here, the cost $C_{cvx} < C_{constraint} < C_{774} < C_{775}$. The choice of the point C_{775} satisfies the budgeted quantization noise power but the cost incurred in the process is higher. On the other hand, the choice of point C_{774} does not satisfy the quantization noise-power constraint.

Formulating the word-length optimization in the convex optimization framework and the use of convex optimization technique ensures global optimality of the solution obtained. However, optimality is achieved only for the relaxed noise-budgeting problem where an imaginary convex Pareto-front is used in optimization. The point of optimality, though not always feasible indicates the locality of the optimal working point. By choosing the word-lengths close to this locality, it is possible to determine the word-lengths that are near optimality. As shown in the previous section, every operator has two feasible points at the most around the point indicated by the convex optimization procedure which need to be considered. Among the two options, it is safe to take a conservative approach keeping in mind the accuracy satisfiability condition. An algorithm which always chooses the conservative choice for word-length determination is presented in Procedure 5.2a.

Procedure 5.2a : GetFinitePrecisionWordlengths()

1: $T(V) = \text{TopologicalSort}(S_i(V, E))$
2: **for all** $v_i \in T(V)$ **do**
3: $\quad db_i = \text{GetOperatorDB}(v_i)$
4: $\quad [Wd_i] = \text{LookupDB}(db_i, q^i_{opt}, \bar{W}d^i_{in})$
5: **end for**

In order to satisfy the input constraints and the propagation of bit-widths across operators, it is important that all operators in the sub-system graph is topologically sorted such that the operators whose input constraints are already known are considered first. By resolving the output fixed-point bit-widths of these operators in turn generate the input constraints for the successive operators. The topological sorting is performed by calling the function **TopologicalSort()**.

The function **GetOperatorDB()** returns the database corresponding to the given operator type. It has to be noted that this database is constructed during the step where the convex Pareto-front of the operator trade-off is determined at the time of constructing the convex noise-budgeting problem. It is a one-time effort for a given

type of operator implementation.

In the function **LookupDB()**, the actual cost of the operator generating the given noise-source with the specified input word-length constraints is determined. In other words, a choice of the actual or most feasible fixed-point configuration is made. Clearly, with any given input constraints, it is possible to propagate the word-lengths across the signal processing system without generating any error from within the system. That is, the fixed-point operator precision is set such that the quantization step-size is not more than the input quantization step-size. This corresponds to 0 noise-power for a given input constraint. If the output quantization step is made suitably larger, quantization noise with a given power can be introduced.

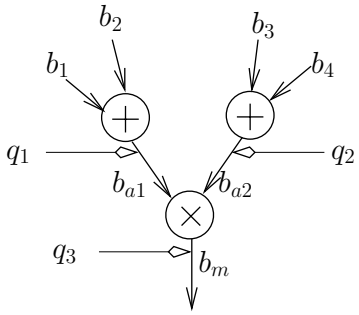
For example, consider a binary multiplier with input word-lengths (a, b). If the quantization noise contribution is assigned a value q, then the number of bits c_{mul} assigned to the output of the fixed-point multiplier is given as

$$c_{mul} = \left\lceil \left(-\frac{1}{2} \cdot log_2 \left(12 \cdot \left\{ q + \frac{2^{-2 \cdot (a+b)}}{12} \right\} \right) \right) \right\rceil \tag{5.32}$$

This is obtained from the quantization noise power contribution as given in Table 2.1. If the quantization noise $q == 0$, then $c = a + b$ which corresponds to bit propagation through the multiplier without causing any more quantization. In case of an adder, Equation 5.32 has to be suitably modified to obtain the quantization.

In the light of Procedure 5.2a, consider determining the word-lengths of the operators graph shown in Figure 5.28. The input constraints to the sub-system are user defined. Thus, the bits b_1, b_2, b_3, b_4 in the Figure 5.28 are user given. The values of the noise contribution q_1, q_2, q_3 of three operators are obtained by solving the convex optimization problem. In order to realise the noise-power q_1, two feasible points q_{b_1,b_2,b_{a1}^l} and q_{b_1,b_2,b_{a1}^h} around the value of q_1 needs to be considered. Here the superscript l represents the greater of the two possible fixed-point word-length assignments possible and the superscript h represents the lesser. In other words, choosing the word-length corresponding to superscript l adds *lower* quantization noise power than budgeted while the word-length corresponding to the superscript h adds *higher* quantization noise power than budgeted. Similarly, to realise noise-power q_2, two feasible points q_{b_3,b_4,b_{a2}^l} and q_{b_3,b_4,b_{a2}^h} are considered. Considering all the combinations of the two inputs, the search space for the multiplier has four different choices at the input. Each of the input combinations can potentially require two feasible points to be considered as shown in the table.

The proposed word-length selection algorithm chooses the combination in the first row of the table as the input bits to the multiplier and chooses the conservative option for the multiplier also. That is, noise-power q_1 is realised using the adder with bit-widths $(b_1, b_2, b_{a1} = b_{a1}^l)$ such that the actual quantization noise power contribution is lesser than q_1. The noise-power q_2 is realised using adder with bit-widths $(b_3, b_4, b_{a2} = b_{a2}^l)$ which is also lesser than q_2. With these choices, the choice for the fixed-point multiplier noise q_3 is realised with input b_{a1}^l, b_{a2}^l and output b_{m1}^l. Under these circumstances, the



b_{a1}	b_{a2}	b_m
b_{a1}^l	b_{a2}^l	b_m^1
b_{a1}^l	b_{a2}^h	b_m^2
b_{a1}^h	b_{a2}^l	b_m^3
b_{a1}^h	b_{a2}^h	b_m^4

$$q_{b_1,b_2,b_{a1}^l} < q_1$$
$$q_{b_1,b_2,b_{a1}^h} \geq q_l$$
$$q_{b_3,b_4,b_{a2}^l} < q_2$$
$$q_{b_3,b_4,b_{a2}^h} \geq q_2$$

Figure 5.28: Propagating bits across noisy operators

total quantization noise contribution by the multiplier is lesser than q_3.

The choice of word-lengths is subjected to input constraints which are determined either by user input of by the word-length assignments of other operators preceding a given operator. If at every operator, a choice of operating point which always generates lesser than the budgeted noise power, the choice is sub-optimal. This effect can easily accumulate over successive decision causing an increase in the cost of the system at the expense of improved accuracy which is anyway not required. On the other hand, if the choice is always such that the smaller word-length is chosen; that is, if the noise-power actually introduced is greater than or equal to the budgeted noise power, the power constraint will not be satisfied. Both extremes are not desirable and the optimal solution lies somewhere in between making such extreme choices. In other words, the choice of word-lengths must be such that some of the choices must offset the effect of remaining word-length choices.

Impact on the Combinatorial Search Space

In the classical word-length optimization, the fixed-point operation is free to take on any of the N word-lengths. So, the combinatorial search space consists of as many as N^m different unique combinations. In the convex optimization framework, the relaxed cost vs. performance curve for every fixed-point operation is used to determine the optimal noise-budgets such that the cost of the system on the whole is minimum. In the *Near-optimal* word-length optimization algorithm, the infeasibility problem of the convex solution is cured by traversing the graph and making suitable choices for fixed-point word-lengths.

As discussed in the previous section, there are two options to choose from in case of each fixed-point operation. Each of the combination of fixed-point word-length of the adders can influence the choice of multiplier word-length and therefore, there can be two choices for the word-length of the multiplier for every adder word-length combination. The table shown in Figure 5.28 show a particular choice for the fixed-point word-length of the multiplier. In Figure 5.29, it is possible to see eight possible unique word-length assignment to the multiplier. This corresponds to 2^3 combinations when there are three fixed-point operations (i.e. two adders and one multiplier). In this figure, the nodes

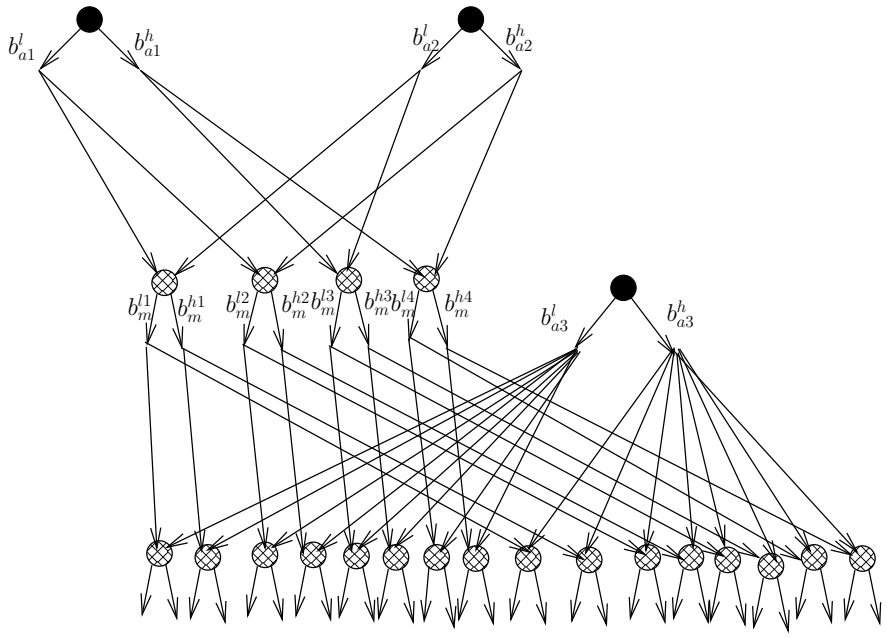


Figure 5.29: Tree representation of all available fixed-point word-length choices

whose input fixed-point constraints are fixed is shown as filled nodes (filled in black) and the shaded nodes indicate the different fixed-point options available. Two choices available for each of the choices are represented by two edges emanating from a node. When another adder and multiplier is added in the computation graph the number of options for fixed-point word-length assignment also increases. It can be seen that the number of options available for the word-length combination of second multiplier can be as many as 32. This corresponds to 2^5 corresponfing to five variables (i.e. three adders and two multipliers).

One of the advantages of using the *Near-optimal* word-length optimization algorithm is that the search spaces shrinks from N^m to 2^m. This shrinkage is due to the binary nature of the choice for the word-length of each fixed-point operation. It has to be noted here that the nature of the problem continues to be exponential in nature inspite of the shrinking in its search space.

Proximity to Optimality

In Procedure 5.2, the choice that can generate quantization noise which is lesser than or equal to the budgeted noise-power is chosen. The choices made is marked along the tree shown in Figure 5.30. In this figure, the word-lengths taken are indicated by taking the left edge at any node. As a consequence of choosing the left edge, the left most node ends up being the choice for all fixed-point operations. On the contrary, if the right edge were to be chosen consistently, the right most node would have been the choices. It is clear that the optimum choice for fixed-point word-lengths lies somewhere

between choosing the left edge or the right edge.

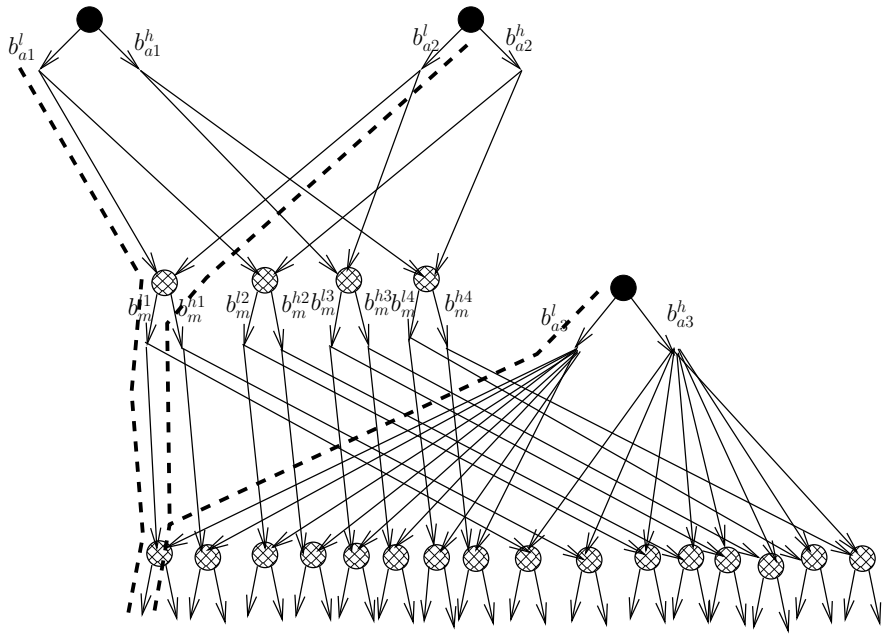


Figure 5.30: Choosing the left-edge always

By choosing the left edge consistently for all fixed-point word-length, it can be said that the actual minimum cost C_{opt} is not greater than the cost $C_{nearopt}$ obtained by following the *Near-optimal* word-length optimization procedure. In other words, $C_{nearopt}$ is an upper bound on the actual minimum cost C_{opt} for all possible choices of fixed-point (in rounding mode) word-length assignments. This is written as

$$\sup_{\mathbf{w}\in\mathbb{N}_+^m} \{C_{opt}\} \leq C_{nearopt}, \tag{5.33}$$

where $\mathbf{w}$ is the vector of word-lengths corresponding to all fixed-point operations whose space is defined m-dimensional and can be assigned any positive integers ($\mathbb{N}_+$). This bound can be improved by using techniques such as *branch-and-bound* while making the word-length choices for each fixed-point variable, In this thesis, the experiments conducted consistently chose the left edge. The improvement in cost obtained over the classical greedy approaches is sufficient to indicate the usefulness of this method.

5.4.6 Results: Convex Optimization vs. Greedy Approach

Signal processing algorithms used in the example of MIMO-OFDM and also the entire system as a whole is considered for the application of the proposed convex optimization framework. The time taken and the quality of results obtained by application of the proposed convex optimization approach is compared with the time taken and the results obtained by the *Min +1 bit* greedy heuristic.

To begin with, in order to apply the proposed convex optimization technique, six types of operators have to be profiled exhaustively. The *Pareto-front* of each of these operators is obtained by the technique described in Section 5.4.3.

Adders and Multipliers

The binary adder and the binary multiplier are most commonly used in the design of signal processing systems. It is imperative to evaluate the *Pareto-front* of these operators for analysis of any signal processing algorithm. The *Pareto-front* of the adder is already shown in Figure 5.25. Similarly, the *Pareto-front* of the multiplier is obtained by exhaustively searching for all points. In the examples considered in this thesis, apart from the multipliers, there are several multiplications that need to be made with constants in algorithms such as FFT. The *Pareto-front* of each for multiplication with constant is arrived at in a similar fashion. This addresses the problem of coefficient quantization and takes into account the number of bits assigned to coefficients. In this thesis, the range of fractional bits assigned to the adder and multipliers is chosen to be between 2 and 24 bits.

CORDIC operators

The CORDIC operator consists of many addition and scaling operations and is also followed by decision. The presence of decision operation makes it difficult to arrive at an analytical formula for the error due to quantization at the output of one CORDIC operation. However, the CORDIC operation is not complex and is easy to simulate. Therefore, it is considered as one of the basic operation and is studied exhaustively and a corresponding noise-power variable is introduced into the convex optimization problem. Further, 32 stages are considered in every CORDIC operation and a uniformly distributed random sequence is used to perform simulation. For various assigned fixed-point word-lengths, the output quantization noise is measured by simulation and its cost is tabulated.

The CORDIC operator is used in two modes. In the vector mode, it calculates the magnitude of the vector represented by the two inputs and determines the angle of the vector. In the rotation mode, a given vector is rotated by the angle obtained from previous vector mode operation. In the experimental setup, the simulation results are compared with the analytical model for performing rotations and vectoring discussed in Section 5.3.4. CORDIC operations in vector mode and in rotation mode for both outputs are exhaustively explored and all the points are plotted in Figures 5.31, 5.32, 5.33. In this thesis, the range of fractional bits assigned to the CORDIC operations is between 2 and 12 bits.

In all the cases, the guiding principle behind choosing the *Pareto-optimal* front is decided by the considerations of optimality as discussed in Section 5.4.2.

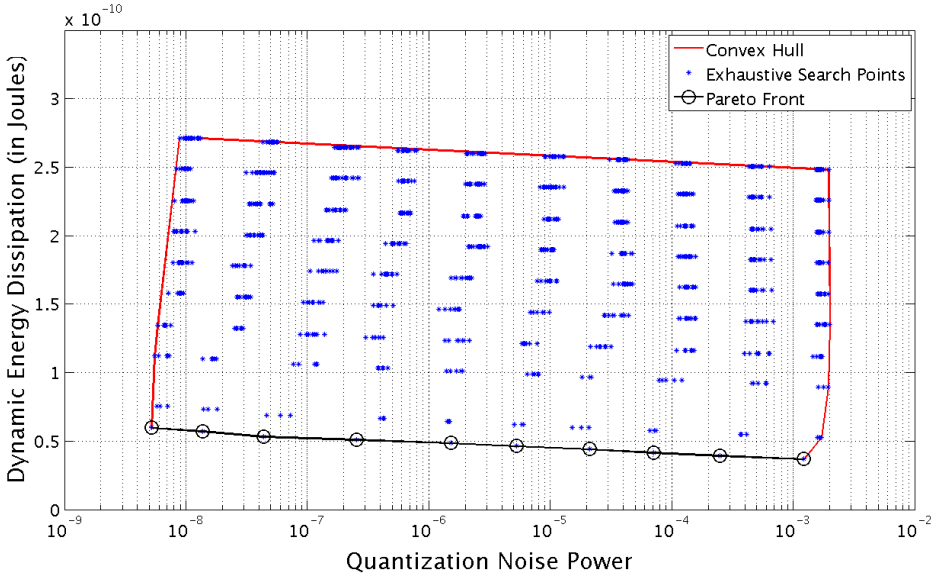


Figure 5.31: *Pareto-front*: magnitude of CORDIC operator in vector mode

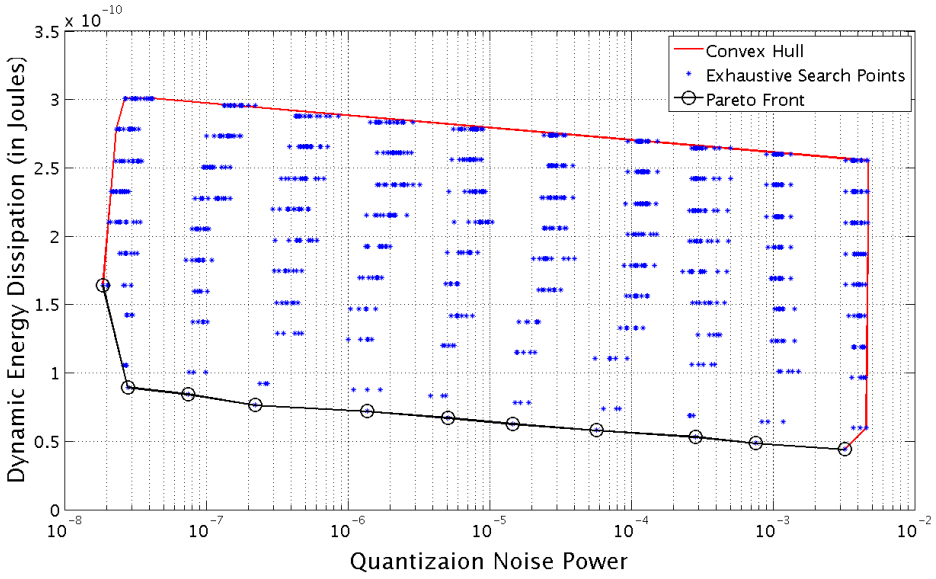


Figure 5.32: *Pareto-front*: x-axis of CORDIC operator in rotation mode

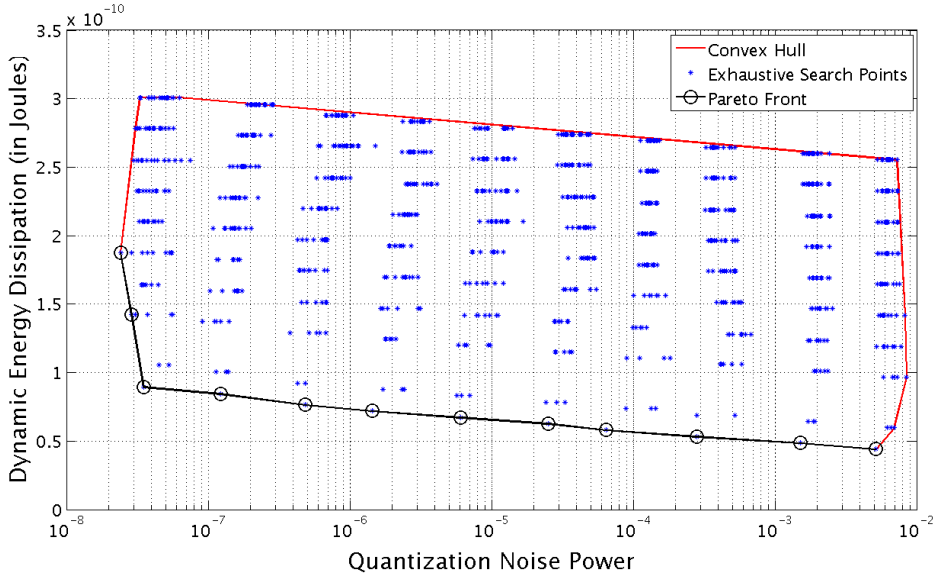


Figure 5.33: *Pareto-front*: y-axis of CORDIC operator in rotation mode

Optimization with CVX

The main task here is to write the routines for evaluating performance and cost evaluation as a function of the vector $\mathbf{q}$ in each case. It has to be written such that an analytical expression can be constructed by the CVX environment as it parses the convex optimization problem. A detailed guide for this coding style is given by the authors of CVX in [48].

Once the optimal noise power distribution is available, the assigned quantization noise is realised using the procedure described in Section 5.4.5. The total quantization noise and the cost is evaluated after all the operators have been assigned a fixed-point format. The results corresponding to four benchmark algorithms participating in the MIMO-OFDM receiver algorithm are presented below.

The cost of fixed-point system obtained in the flat approach and by the application of convex optimization framework can be observed in Figure 5.34. Clearly, the improvement in the over all cost increases with the system size in case of the FFT algorithm. In case of the QR algorithm, the cost improvement is achieved but to a lesser extent.

The use of *Pareto-front* for optimization and then searching for the fixed-point solution in the locality of the solution is the main reason for achieving the cost improvement. The greedy heuristic always latches on to a feasible point in every iteration. Some suboptimal choices made during early iterations can cause a cascading effect and it may so happen that none of the choices presented is optimal in the later iterations. Since the direction of optimization is always towards incrementing the bits assigned, the optimization process may never recover from suboptimal choices made during early iterations. On the other hand, in the convex optimization approach, the *Pareto-front*

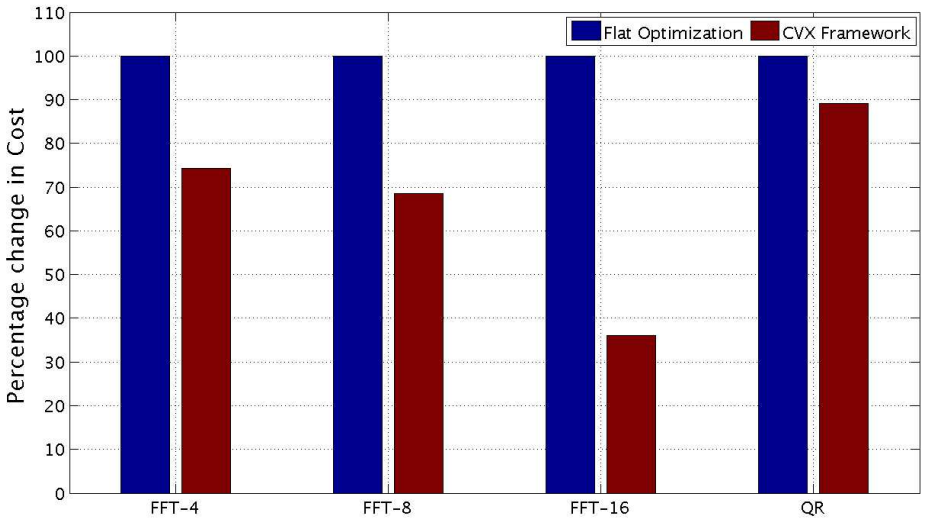


Figure 5.34: Comparisons: cost of implementation

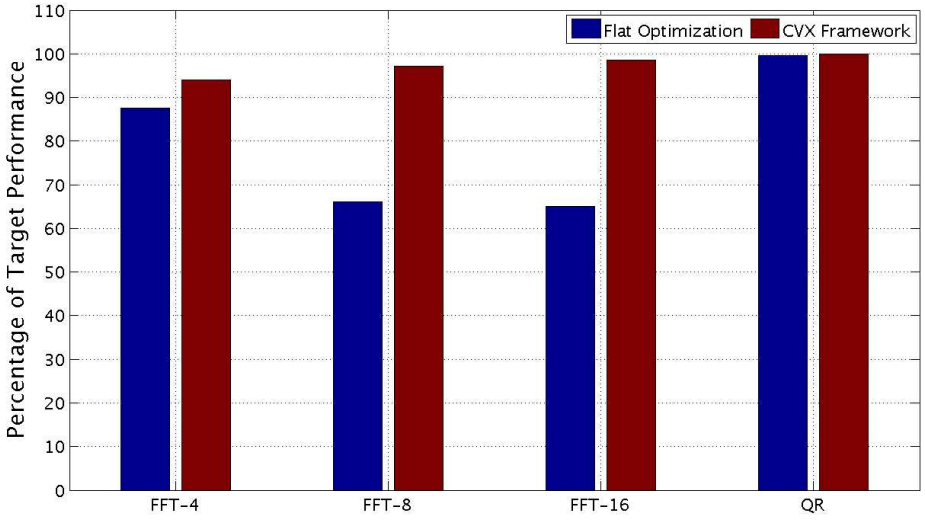


Figure 5.35: Comparisons: performance achieved

is used and the actual fixed-point determination is carried out as a final step. By then the actual quantization noise power from each of the source is well established. It has to be noted here that if fractional word-length assignment were to be possible, then the solution obtained by the convex optimization problem solver is truly optimal. The sub-optimality factor creeps into the proposed convex optimization framework when

the noise power is realised as fixed-point operators. As suggested in Section 5.4.5, a *branch-and-bound* algorithm can further reduce this sub-optimality and can potentially take the solution much closer to optimality.

The actual quantization noise achieved by both the processes are shown in Figure 5.35. The desired quantization noise power generated by the fixed-point design is set to 100%. The actual quantization noise power is achieved and it is usually lower than the desired noise power. In all the cases shown in Figure 5.35, the quantization noise power target is nearly reached by the convex optimization framework. Whereas, the classical approach tends to over-optimize. This is due to the fact that the classical approach moves from one fixed-point format to the next and thereby accruing sub-optimality in every iteration.

5.4.7 Optimizing MIMO-OFDM Receiver

The complete functional block diagram of the MIMO-OFDM Receiver algorithm is given in Figure 5.41. Although the results of the *FFT* and *QR* decomposition can be shared between different smooth blocks of the *V-BLAST* sub-system, they can be functionally separate. That is, these sub-systems can have different fixed-point implementations. For the sake of clarity, the dependency between the output of un-smooth operators and the smooth blocks is not marked explicitly. It is easy to follow this dependency by following the labels given to these signals. In the previous section presenting the results of the convex optimization framework, the V-BLAST example was not presented. This is due to a practical limitation with the tools used for evaluating the proposed convex optimization approach. The use of the convex approach using the CVX package is limited to those scenarios where the performance and cost evaluation functions can be expressed analytically. The presence of un-smooth operator makes it impossible to code both the hybrid simulation approach or the analytical approach for performance evaluation. Therefore, the convex framework implemented using the CVX package could not be applied to the V-BLAST algorithm. It should be noted here that there is not such limitation of this nature in theory. It is just that the developing a convex optimization solver which can be used generically is out of the scope of this thesis work. In this thesis, an alternative approach which uses the greedy heuristic presented in Section 5.2 and the convex framework are used together to optimize the V-BLAST algorithm and also the entire MIMO-OFDM receiver.

MIMO-OFDM Receiver: Hierarchical Decomposition

The functional sub-systems of the MIMO-OFDM receiver were studied in Section 5.3. The global problem is that of word-length optimization of the MIMO-OFDM receiver algorithm. The receiver is decomposed into four sub-systems at the QAM slicer boundaries corresponding to the estimation of the symbol received by each Antenna. To arrive at this estimation, it is required to calculate the FFT from all four antennas and to evaluate the QR decomposition and execute the corresponding block from the V-BLAST block diagram. These sub-systems within each of the antenna blocks are

marked FFT_1 through FFT_4 corresponding to the FFT after each antenna, QR to perform the QR decomposition and INV_1 through INV_4 to perform the V-BLAST blocks respectively. The hierarchical decomposition of the receiver algorithm is as shown in Figure 5.36.

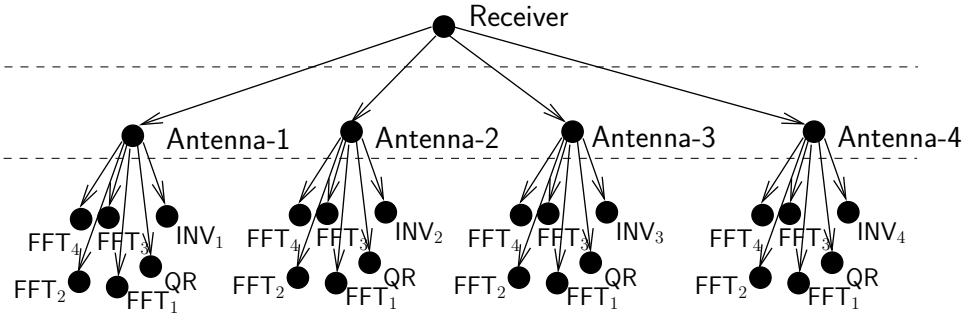


Figure 5.36: MIMO-OFDM receiver: hierarchical decomposition

Each of these sub-system blocks can be sub-divided further as discussed in Sections 5.3.1, 5.3.2, 5.3.3. Hence, they are not shown as leaf-level sub-systems. In a parallel hardware implementation, the hardware corresponding to this decomposition consists of 16 FFT operations and 4 QR operations. However, it is unlikely that such redundancy is ever provided in hardware. Therefore, it is assumed that there is one unique FFT block processing the input received by each antenna and the QR decomposition module is shared across all 4 antennas. The hierarchical decomposition after applying the resource sharing constraint results in the hierarchical decomposition as shown in Figure 5.37

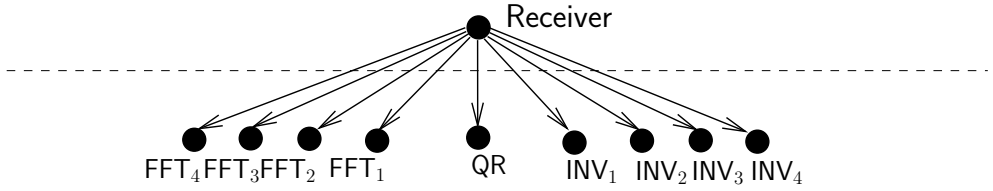


Figure 5.37: Hierarchy after considering resource sharing by FFT and QR modules

The optimization problem at the global level has 9 optimization variables. While the optimization of each of the sub-systems can be carried out using the convex-optimization framework, the global optimization requires the use of the greedy heuristic discussed in Section 5.2. The 3 dB step-size was chosen to carry out the adaptation of the greedy heuristic.

The target performance set was a degradation of 5% in BER of the receiver over the BER obtained in double precision. The evolution of the total cost starting from the maximum noise power until the constraint is satisfied is shown in Figure 5.38. To understand the numbers better, the cost of each sub-system and the overall cost is normalized to 100%.

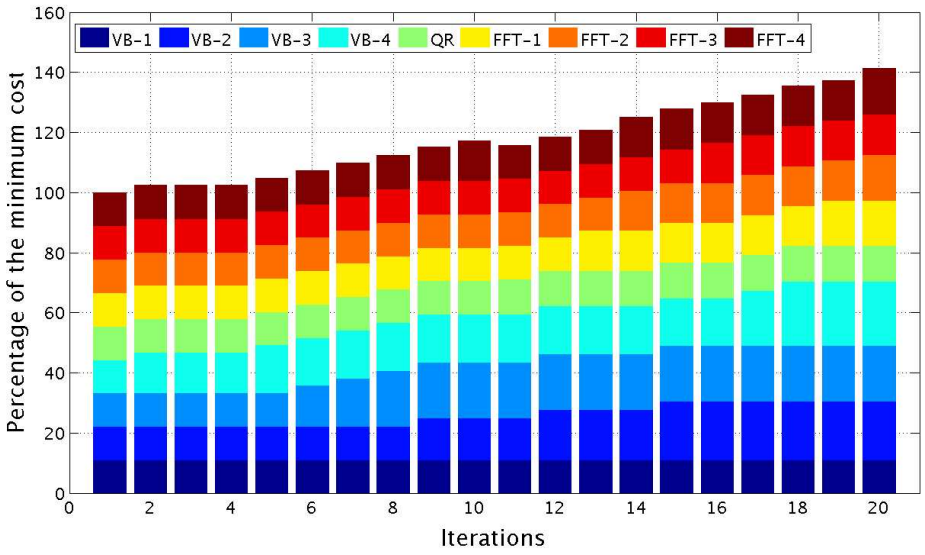


Figure 5.38: Relative cost of the sub-systems in every successive iteration

There is about 40% increase in the cost to achieve the targeted 5% degradation in BER with respect to the initial (minimum) cost. In terms of relative cost, consider the number of arithmetic operations in each block. Considering 32 rotations and two operations per rotations, there are 18 such modules present in the signal flow graph of QR decomposition. In contrast, there are $64 \times log_2(64)$ additions and $64 \times log_2(64) \times 0.5$ constant multiplications that are realised using additions with the FFT. The V-Blast algorithm has an average of 10.5 operations per antenna block. Therefore, the QR module consumes the maximum energy followed by the FFT blocks and then the V-BLAST blocks. This is also reflected in the actual final costs of each of the blocks as shown in Figure 5.39 .

The process of optimization takes 20 steps and the evolution of the trade-off between the cost and the symbol error rate is shown in Figure 5.40. In the 10^{th} iteration, the choices made cause retrograde in the advancement of the optimization. Due to the discrete and non-convex nature of the combinatorial iterations, several undesired scenarios can occur during the course of iterations. It is possible that for some of the options, a change in the total cost does not result in any change for the total quantization noise power. Due to the influence of the un-smooth operator, it is also possible that some of the options do not show a change in the BER. Under such circumstances a random step is chosen which can result in such retrograde steps. The effect of this retrograde step is also reflected in the relative cost graph in Figure 5.38 between iterations 10 and 12.

With the help of the CVX based approach, the MIMO-OFDM receiver was optimized in about four hours time in Matlab running on an Intel based MacBook Pro from Apple [57]. Due to the complexity of the complete system, it was not possible to carry

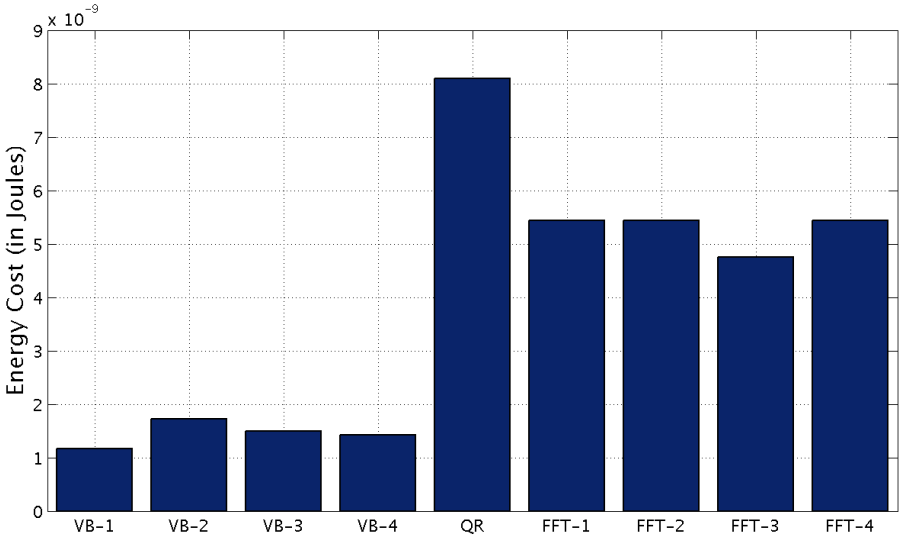


Figure 5.39: Cost of individual sub-systems after reaching target SER (Symbol Error Rate)

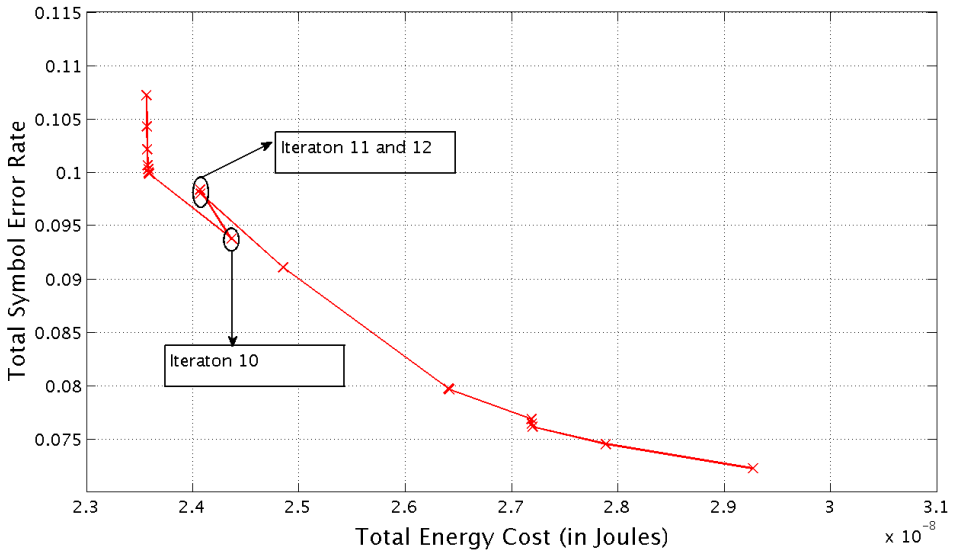


Figure 5.40: Evolution of SER vs. Cost trade-off

out the word-length optimization using the flat approach.

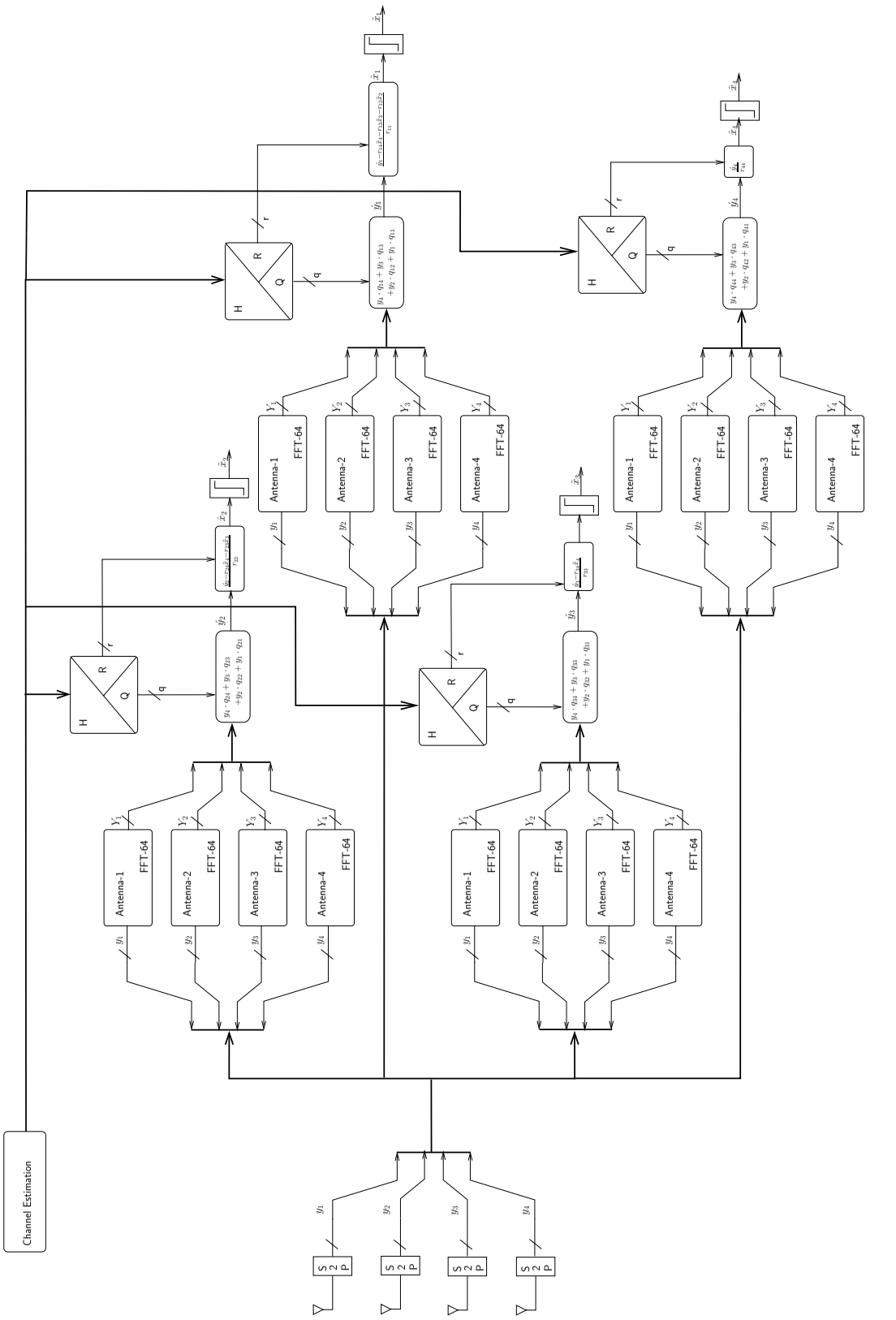


Figure 5.41: MIMO-OFDM Receiver: Functional Block Diagram

5.5 Summary

The cost minimization by choice of appropriate word-lengths of fixed-point operations is considered. Large word-length optimization problems present a huge combinatorial search space in which the optimal working point is contained. Searching the entire space for finding out the optimal working point is daunting and virtually impossible. Therefore, popular solutions use heuristics hoping to find the optimal working point. Thus far, there has been no claim or proof for the optimality of the techniques.

With growing system sizes, the complexity of the word-length optimization problem becomes unmanageable with any of the existing techniques. This is due to the scalability issue with existing algorithms. In order to continue to use the existing heuristics, a divide-and-conquer method for solving the cost minimization problem is proposed. The proposed hierarchical technique utilizes the total noise-power contribution at the output of every sub-system due to fixed-point operations as the optimization variable in this process. The number of iterations required for solving large word-length optimization problems is reduced by several orders of magnitude by adopting the proposed hierarchical method while the quality of the results remains comparable to what would have been obtained by using a classical flat approach.

The sub-optimal nature of the greedy word-length optimization algorithms is discussed. It is observed that the non-convexity of the trade-off between the number of bits and the cost is the reason for the under-performance of the greedy algorithm. A novel technique which makes use of principles from the convex optimization framework is proposed to perform the word-length optimization. In this technique, several relaxations and assumptions are made such that the word-length optimization problem is rendered convex. A *near-optimal* word-length assignment technique is proposed to translate the results obtained by solving such a convex optimization problem onto realisable fixed-point solutions. This technique uses the popular *CVX* toolbox in Matlab for the purposes of optimization. This complexity of this technique to solve the word-length optimization is polynomial in time. Also, the results obtained outperforms the results obtained by using the popular $Min +1bit$ algorithm. Therefore, the proposed framework is not only faster but also generates better quality results.

The proposed framework has a practical limitation which strictly requires analytical functions for the performance and cost evaluation functions of sub-systems. This limitation is imposed by the use of *CVX* toolbox. The advantage however is that the solution is obtained very quickly due to analytical functions. In this thesis, the OFDM-MIMO receiver algorithm is chosen to demonstrate the efficacy of the proposed optimization techniques. This algorithm has several hundreds of fixed-point operations and is a classic case which demonstrates the scalability of classical word-length optimization techniques. Here, the hierarchical word-length optimization is applied at the top-most level and the novel convex optimization approach is applied for sub-systems in subsequent levels. The word-length optimization of this algorithm takes about four hours when the proposed optimization techniques are applied in the Matlab environment.

Chapter 6

Conclusions and Future Work

The use of fixed-point platforms has been there since very early days of computers. In fact, they were the first circuits to be used for computation when digital computers were first designed. Limited by the technology of the day, to realise fixed-point operations using electronic circuits itself was a formidable challenge. With advancing silicon process technology, the transistor sizes shrunk and paved way for more complex digital circuits enabling the innovation of floating-point units. Although the floating-point format was designed and has been standardized since many decades, the fixed-point circuits continued to be used due to the speed and simplicity advantage they provided. The design of fixed-point algorithms essentially focusses on trading off dynamic range and precision to the total implementation cost such that the computational performance is not compromised.

In the recent years, fixed-point arithmetic circuits have been increasingly used for the implementation of signal processing systems on embedded systems. Design of portable electronic devices imposes severe constraint on the power profile of electronic devices. Efforts for power conservation in particular has been the focus of industry and academia in the recent years. While several popular power optimization techniques such as power-gating, voltage frequency scaling etc. have mostly been at the circuits level, a new dimension to this problem is added by choosing to optimize energy by using suitable fixed-point formats.

The problem of word-length optimization or fixed-point refinement of signal processing algorithms has caught the attention of many research group over the past decade across the world. There are essentially two paradigms for word-length optimization. They are the uniform word-length (UWL) optimization paradigm and the multiple word length (MWL) optimization paradigm. While the former design approach explores a limited number of options, the latter proves to be more effective as it essentially reaches out to a large design space. In this thesis the MWL optimization paradigm is addressed. This problem is combinatorial in nature and is known to be NP-hard. Several techniques based on greedy heuristics and genetic algorithms have been suggested in the past for performing word-length optimization. This problem in general essentially consists of three sub-aspects: i) the problem of dynamic range evaluation, ii) the problem

of accuracy evaluation, and iii) algorithm for word-length optimization. Solving the last two problems among these three sub-problems has been the focus in this thesis.

The ubiquitous use of fixed-point operations requires study of algorithms in various domains. Broadly, the algorithms implemented using fixed-point platforms can be modeled as either pure data-flow-graphs (DFG) or control-data-flow-graphs (CDFG). The presence of control signals among many data signals makes the flow of data indeterministic. Although the treatment of CDFG class of algorithms is in the best interest to this field, the indeterministic nature of data flow in the presence of control makes it difficult for application of analytical techniques. On the other hand, a large class of signal processing algorithms can be modeled using pure DFG structures. Keeping this in view, the focus of this thesis narrows down to fixed-point refinement of signal processing algorithms.

Although this thesis only partly addresses the fixed-point refinement issue, the target class of algorithms to which the proposed techniques can be applied to has a huge impact on the way modern electronic systems are designed. Using signal processing implementations on resource constrained platforms poses a challenge in several important technological domains such as wireless or wireline communication systems, multimedia, entertainment systems etc. Here, all the contributions made in this thesis are briefly reviewed and the scope for future work is discussed.

Summary of Contributions

Contributions in this thesis have been presented in the previous three chapters. In this section, key ideas in this thesis that have paved way to solve some interesting challenges are summarized and an attempt to provide a high level perspective to the contributions is made.

Stochastic Modeling of Quantization Noise

The first contribution made in this thesis is inspired by the idea behind the pseudo quantization noise (PQN) model. The applicability of PQN model is one of the fundamental assumptions made during linear analysis of quantization noise in a fixed-point system. Quantization is by definition, non-linear due to the discontinuity. However, it is still amenable to linear analysis when it satisfies the conditions required set by the PQN model (also referred to as quantization theorems). All instances when fixed-point operations satisfy the conditions for the application of PQN model are referred to as smooth operations. The PQN model essentially models the fixed-point errors as a particular random process which is spectrally white, uncorrelated with the signal and has a uniform distribution whose mean and power are determined by the quantization step-size.

While the PQN model addresses the characteristics of quantization noise generated at the output of a fixed-point quantizer, the single noise source model (SNS) scales up this idea to be applied to abstractions of fixed-point systems at large. The SNS model

is essentially about deriving the characteristics of errors due to fixed-point quantization operations at the output of a given system. The usefulness of this approach is that it enables the treatment of large signal processing systems at sub-system level abstractions rather than requiring to deal with individual fixed-point operations. This has a profound impact on the scalability of techniques used for fixed-point refinement.

The SNS model essentially consists of three aspects: i) total quantization noise power, ii) spectral distribution of noise power and its spectral correlation with other signals, iii) the noise shape describing its spatial distribution (PDF). While the determination of quantization noise power has been addressed in previous works, this thesis presents analytical expressions for deriving the spectral shape and noise PDF. With relevant examples, the efficacy of using these parameters for modeling fixed-point effects is demonstrated in Chapter 2.

Although a one-time effort, the complexity of deriving each of the SNS model parameters can be high depending upon the system functionality. However, it is enough to derive only the relevant SNS parameters are derived. In case of systems that are not sensitive to either the signal level or the signal frequency, it is sufficient to work with only the noise power parameter.

Performance Evaluation with Unsmooth Operations

The smooth quantization operations which behave according to the PQN model tend to be more accurate when the step-sizes are smaller. When the quantization step-size is large enough, the behavior suggested by PQN model is no longer valid. Under such circumstances, quantization is said to be un-smooth. This is essentially due to the fact that the signals in practice are time-limited and are therefore band-unlimited in the characteristic function domain. Depending upon the accuracy of estimation required by the quantization noise itself, it is possible to define a boundary for every given quantization operation to say if it is smooth or un-smooth. An algorithm to perform this classification is also presented.

Apart from quantization with large step-sizes, other operations such as *Min()*, *Max()*, QAM slicers and essentially any other operation which does not have a linear noise propagation model is considered un-smooth. In the presence of un-smooth such as a QAM slicer, a fully analytical technique for evaluating quantization errors is presented. This method requires the knowledge of the PDF of both signal and noise at the input of every un-smooth operation. The evaluation of error due to fixed-point is made by essentially propagating the errors due to fixed-point operations through various operations. Two separate cases where the decision outputs from QAM slicers may be correlated or un-correlated with one another are considered. This technique is applied for analysis of the errors at the output of various QAM-slicers in the V-BLAST algorithm.

The strategy for propagation of quantization noise through un-smooth operations fails when the system is recursive in nature. Another alternative to the rather time consuming analytical approach which can also work in the presence of decision feed-back loops in the algorithm is proposed. This is referred to as the *Hybrid* approach. This

approach is the first of its kind to use analytical techniques for fixed-point accuracy evaluation for accelerating fixed-point simulation. This technique selectively simulates parts of the system only when an un-smooth error occurs. The given system is divided into many smooth clusters consisting of many sub-systems with smooth quantization operations. These smooth clusters are separated by un-smooth operations. The application of SNS model on each of the smooth clusters captures the statistical behavior of quantization noise accurately. This characterization is used to generate random numbers to mimic quantization errors rather than perform fixed-point simulation. Thereby, avoiding wasting precious time simulating fixed-point numbers. This technique also involves book keeping of un-smooth errors for switching between simulation and analytical evaluation modes.

Several examples including the decision feed-back equalizer are studied in this context. This technique is essentially based on floating-point simulation and the use of SNS model. It yields several orders of magnitude improvement in the time taken for fixed-point simulation even while being statistically consistent with fixed-point simulation.

Hierarchical Approach

The scalability of the fixed-point refinement process essentially depends on both, the functional complexity of the algorithm and the degree of freedom available for the choice of word-length assignment. Two scalability problems that arise as the system grows in its size are addressed in this thesis.

The first among them is the accuracy evaluation in fixed-point systems. The *evaluate-propagate-add* strategy discussed in Section 3.1.2 addresses this problem with the help of the SNS model. In this approach, the given system is functionally decomposed into several sub-systems with reduced functional complexity. The SNS model is used to evaluate the quantization at the output of each of these sub-systems. This makes it possible to explore opportunities for parallelize evaluation of fixed-point noise at the system output. Moreover, when the word-length assignment to any of the operations changes (say, during iterative optimization), the evaluation of output quantization noise power can be performed by evaluation and propagation of output quantization noise for only those sub-systems where the word-length has been changed.

The second problem is the scale of scalability of the word-length optimization algorithm. As the number of variables participating in the optimization problem increase, the optimization process takes longer to converge. The number of variables depends both on the number of operations and the degree of freedom available to make fixed-point word-length choices. For example, there is only one degree of freedom when the uniform word-length (UWL) paradigm is used and the number of degrees of freedom increases as the operators are allowed to have different choices of word-lengths. Ideally, it should be possible to assign each fixed-point operation a word-length by uniquely considering their impact. Therefore, to consider that there are as many degrees of freedom as the number of fixed-point operations is idealistic.

In this thesis, a hierarchical approach for addressing the increasing complexity of

the word-length optimization problem is devised. In this approach, the sub-system boundaries are identified by the functional decomposition as it is done for accuracy evaluation. The degrees of freedom for assignment of word-lengths is determined by considering the resource sharing constraints. A bottom up approach using the well known *divide-and-conquer* strategy is proposed for hierarchical optimization of the given system.

Several examples: sub-systems obtained from a MIMO-OFDM receiver system is used for evaluating the impact of hierarchical decomposition on the word-length optimization problem. The results indicate a reduction in the number of iterations by at least one order of magnitude. The actual time taken including the sub-problem optimization takes significantly lesser time than the flat optimization approach.

Given the NP-hard nature of the word-length optimization problem, the heuristics suggest an exit clause to terminate iterations. When the iteration terminates, it is not necessarily true that the assignment of word-lengths is optimal. This is due to the non-convex behavior of the cost and accuracy trade-off. In this thesis, a convex optimization based framework is proposed for solving the word-length optimization problem. The combinatorial problem is transformed to an equivalent convex optimizatoin problem by relaxing the integer constraint on the word-length assignments. Further, the fixed-point design space of each type of operation such as a basic operator such as a binary adder is explored exhaustively. The search space is characterized with a scatter plot of cost vs. accuracy for each feasible fixed-point design. A *Pareto-front* is constructed by choosing a part of the convex polygon covering all the feasible fixed-point design points explored. This *Pareto-front* is convex (by definition) and is used as the trade-off instead of the actual feasible points during optimization process. The transformed problem is now called the *noise-budgeting* problem. Here, the objective is to distribute the total quantization noise among different operator level noise source optimally. The *CVX* toolbox in Matlab is used to solve the *noise-budgeting* problem thus formulated. The solution obtained by this process suggests the quantization noise that can be injected into the system at the point in the system graph corresponding to the fixed-point operator under consideration.

The solution obtained by solving the *noise-budgeting* problem uses the relaxed version of the original problem. Therefore, the solution is only indicative of the locality of the solution and not by itself the actual solution. A *near-optimal* word-length assignment algorithm presents a technique which essentially searches for nearby points that can be realised using fixed-point operations with integer word-lengths. The *near-optimal* word-length assignment algorithm essentially checks for just two possible word-length values that can be assigned to each fixed-point operation. There by the search space reduces from the original case where any arbitrary word-length assignment were to be possible. Using the convex framework for word-length optimization, it has been possible to define the supremum value of the minimum cost by always taking the conservative choice.

Perspectives

In this thesis, the scalability issues with respect to the fixed-point performance evaluation and word-length optimization have been the main drivers. The contributions made in this thesis have been able to diligently address them and propose workable solutions to address these issues. The three aspects that have been addressed in this thesis aid one another in improving the time and quality of the fixed-point refinement process.

A MIMO-OFDM receiver algorithm described in Section 5.3 is considered. This choice is repersentative of the typical signal processing algorithms that are being deployed on silicon in the present day consisting of thousands of fixed-point operations. The use of analytical techniques that existed prior to the contributions in this thesis is not possible for as they are not applicable in the presence of un-smooth operations. Therefore, fixed-point simulation has to be inevitably used for evaluating the impact of fixed-point operations on this system. The performance evaluation by fixed-point simulation of such a system consisting of thousands of fixed-point variables takes a very long time.

The application of *evaluate-propagate-add* helps in evaluating the performance of the system by evaluating the output quantization noise of smaller sub-systems. However, the applicability of this approach is limited to the FFT and QR sub-systems due to the presence of un-smooth operations in the V-BLAST sub-system. It is possible to accrue the quantization errors up until the input of V-BLAST and only the V-BLAST sub-system requires to be simulated. This is made possible as the quantization noise properties can be captured fairly accurately by application of the *single-noise-source* models at the output of FFT and QR sub-systems. The *Hybrid* approach pitches in at this point to improve this process further by improving the time taken for fixed-point simulation of the V-BLAST sub-system by over two orders of magnitude over fixed-point simulation in the presence of un-smooth operations. Thus, the different techniques proposed in this thesis can improve the time taken for performance evaluation of fixed-point systems. This essentially means that by using the techniques proposed in this thesis, computers can now take on larger systems than what was possible by the classical flat analytical approach or fixed-point simulation based approaches for performance evaluation.

The large number of fixed-point optimization variables poses a challenge during the actual word-length optimization process. If each of these fixed-point operations have to be assigned a word-length by due consideration of their impact on the output, it is virtually impossible with classical word-length optimization techniques. The *Hierarchical* technique for word-length optimization presented in this thesis provides a *divide-and-conquer* approach for word-length optimization of such large problems. The total number of optimization variables is greatly reduced due to the hierarchical decomposition of the problem. The sub-problems are simple enough to be optimized with classical techniques. The number of iterations required for optimization of these sub-systems reduces by several orders of magnitude. In case of the V-BLAST sub-system, a two orders of magnitude reduction is obtained. Indeed, in case of just the

V-BLAST sub-system case, a reduction in time of nearly four orders of magnitude has been achieved by using both the *Hybrid* technique for performance evaluation and the *Hierarchical* technique for word-length optimization.

The *Convex-optimization* based framework for word-length optimization is not only generates solutions that are better in terms of its quality but are also faster than the iterative hierarchical optimization technique. The total optimization time for the entire MIMO-OFDM receiver takes only about four hours on an Apple Mac-book PRO platform for assigns word-lengths to each of the thousand fixed-point operations giving them the due consideration. Optimization of such a huge system is almost unimaginable in practice without the use of the convex optimization and hierarchical techniques proposed in this thesis.

Future directions

The thesis centrally focuses on the problem of scalability by addressing system-level issues. While some ideas explored in this thesis ended up solving the problems that were intended to be solved in the first place, some assumptions that were made in the course of this study need to be questioned again. In this section, limitations of the proposed solutions are explored and some of the key assumptions are questioned to define possible scope for future work.

The single noise source (SNS) model is derived and used in this thesis for studying the average case performance of fixed-point systems. This could also be used for studying the worst case/ corner case scenarios. The quantization noise PDF at the operator source is known from the PQN model. In order to make the worst case analysis, it would just be enough to propagate the PDF to that point of interest in the signal flow graph. This needs to be compared with the traditional PDF propagation approaches based on methods such as PCE[1]. As the proposed PDF propagation by evaluating Kurtosis works well, it is likely that this method will simplify the evaluation of worst case scenarios at every point in the signal flow graph.

The convex optimization framework requires zero mean noise sources for the optimization problem to work. It is required to extend this to include the case when quantization noise sources have non-zero mean. The *near-optimal* word-length optimization algorithm uses a conservative approach. A *branch-and-bound* algorithm can be used instead to take the solution closer to the to optimality. Secondly, the effect of hierarchical grouping of sub-systems on the quality of convex optimization approach. Further study is required to understand this.

This thesis uses a simplified energy cost model which provided a good first cut estimate on the system implementation cost. However, it does not capture several artefacts during a semi-custom design flow on hardware. Several decisions taken during various stages such as scheduling, binding and place and route has a profound impact on the performance of the circuit. There can be cases in practical circuits where the extra bits added to a particular operator required a long routing making the system

[1] Polynomial Chaos Expansion

effectively slow. In such cases, removing some of the bits might speedup the circuit without compromising too much on accuracy. Therefore, it will be worth trying to formulate a joint optimization problem to solve some of these problem together.

While the previous suggestion refers to a specific kind of problem explanation, what is essentially required is a more specific cost model. In this thesis, the cost is just a function of number of bits. A high-level model for capturing the cost metric will can have a huge impact on the quality of the result obtained eventually. This requires modeling of some of the algorithms used down the chain in the high-level synthesis (HLS) tool flow in order to abstract away relevant artefacts for the purposes of use during word-length optimization.

The hierarchical decomposition and resource sharing has a profound impact on how quickly a given algorithm can be solved. The decision to share resources between subsystems is user defined. This has close connections to the resource binding problem in the HLS tool flow. With prior information about binding, it is possible to influence the time taken for optimization. At the same time, by choosing to have a lot of independence, it is possible to influence the binding phase.

It is quite an engineering effort to realise the various ideas presented in this thesis as a tool chain. Such a tool can indeed be a part of a larger support high level synthesis tool. The ID.Fix project, a part of the GeCoS framework is in its early stages of development. This ideas generated during this thesis is intended to be a part of the ID.Fix infrastructure for automatic fixed-point refinement.

Appendix A: Selective Evaluation of Spectral Parameters

In this appendix, algorithms for performing selective evaluation of spectral parameters of the Single Noise Source (SNS) model discussed in Section 3.5 is presented. The spectral parameters are evaluated selectively by taking the frequency sensitivities of various sub-systems into consideration.

Consider a directed acyclic graph $H(V, E)$, which captures the connectivity between the leaf-level sub-systems. The procedure to do the selective evaluation is as shown in the procedure A.1. Let *GetOutputNodes(*$H(V, E)$*)* be the function which returns a list of output nodes. Likewise, let *GetPaths(*H, i, j be a function which enumerates all the paths between any two nodes V_i, V_j in the graph $H(V, E)$. This function returns an exhaustive list of paths which connect any two nodes V_i, V_j in the SFG $H(V, E)$. If the two nodes are connected, there can be one or more paths and the function returns *null* if the two nodes are not connected. In this procedure, all paths $\bar{P}_{ij}$ between the i^{th} sub-system and the j^{th} nodes are enumerated. In a connected system graph, some of the sub-system outputs may not be connected to all the outputs. For example, in the graph shown in Figure 3.21, S_2 is not connected to o_2. Therefore, the path vector has to be checked for *null* before proceeding.

Procedure A.1 : Selective Evaluation of Spectral SNS Parameters

```
1: Initialize(H(V,E));   \* Initialize the data structure to default behavior *\
2: V̄_o = GetOutputNodes(H(V,E));   \* returns {o1, o2} for example 3.21 *\
3: \* Search all paths from every node to every output in the SFG *\
4: for all nodes: V_i in the graph H(V,E) do
5:    for all nodes: V_j ∈ V̄_o  do
6:       P̄_ij = GetPaths( H(V,E), V_i, V_j);
7:       if  P̄_ij ≠ 0  then
8:          AnnotateACorrParams(P̄_ij, V_i, V_j, H(V,E));
9:          AnnotateXCorrParams(P̄_ij, V_i, V_j, H(V,E));
10:      end if
11:   end for
12: end for
13: EvaluateSpectralParams();
```

The spectral parameters can be divided into two broad categories. One set of parameters required for determining the autocorrelation power spectrum and the other defining the cross correlation spectrum between outputs of any node considered in pairs. Once all the paths from a given node to the output are enumerated, the functions *AnnotateACorrParams()* and *AnnotateXCorrParams()* are called to mark the required auto correlation properties and cross correlation properties respectively. These procedures check if the autocorrelation or cross-correlation properties require to be marked for every node that lies along every path.

Once the nodes are annotated with the spectral information that needs to be evaluated, the function *EvaluateSpectralParams()* evaluates the expressions for the autocorrelation and cross correlation spectrum of those signals which are marked during the annotation process parameters as discussed in Section 3.3.

Annotating Auto-correlation Parameters

The number of paths between sub-system V_i and output V_j in the vector $\bar{P}_{ij}$ is at least equal to the out-degree of the node V_i. The procedure A.2 presents an algorithm to identify the sub-system nodes in a given SFG for evaluation of auto-correlation power spectral density.

In the graph shown in Figure 3.21, consider the paths from node S_1 to output o_1. There are two edges fanning out of the node S_1 and hence there are two paths from the node S_1 to node o_1. One, which is along the edge e_{12} and the nodes S_2 and S_3 and the other which is along edge e_{14} and the nodes S_4, S_2 and S_3. So, k can take on two values in this example.

The contribution of any operator noise source g to the sub-system output is non-white only if the corresponding path to the output is frequency selective. Even a single path from any of the operator noise sources to the k^{th} output that containing memory elements makes the path frequency selective. Therefore, $C_k(\nu, e)$ which is a graph consisting of all elements in the sub-system belonging to the fan-in cone of the k^{th} output of the node V_i is extracted. The presence of memory elements in $C_k(\nu, e)$ makes it imperative to evaluate the auto-correlation function of the k^{th} output. This corresponds to the *Generate Filter* in the single noise source model as shown in Figure 3.7.

The noise generated at the k^{th} output propagates along p_{ij}^k all the way to the j^{th} node. In each node that is encountered along this path, the actual transmission path q_{ik} in each of nodes is added to the *TransmitCorrelation* list of the corresponding nodes. The input output signal pairs marked in the *TransmitCorrelation* list defines the *Transmit Filter* component of the SNS model of the sub-system of the corresponding node.

Annotating Cross-correlation Parameters

The cross-correlation parameters need to be evaluated if there are multiple outputs emanating from the same noise source and they re-converge later in the graph. This condition is detected by checking for intersection among the paths between any two

Procedure A.2 : AnnotateACorrParams($\bar{P}_{ij}, V_i, V_j, H(V,E)$)

```
1: for all Paths: p_ij^k ∈ P̄_ij do
2:     \* Marking spectral characteristics for Generate Filter *\
3:     \* Get the k^th output Cone *\
4:   C_k(ν, e) = GetOutputCone(k, V_i);
5:     \* Check if there are memory elements in the Cone *\
6:   if Memory element ∈ C_k(ν, e) then
7:     V_i.GenAutoCorrelation += k;
8:   end if
9:     \* Marking spectral characteristics for Transmit Filter *\
10:  \* Go to the next node in the path p_ij^k *\
11:  V_i = GetNextNode(p_ij, V_i);   \* The next node is now V_i *\
12:  while V_i ≠ V_j do
13:    V_k = GetNextNode(p_ij, V_i);   \* Get the node next to V_i *\
14:      \* Get path in the sub-system V_i between input i and output k *\
15:    q_ik = GetSubsystemPath(V_i, i, k);
16:      \* Checking if this path is frequency selective *\
17:    if Memory element ∈ q_ik then
18:        \* Transmit Filter of V_i needs evaluation *\
19:      V_i.TransmitCorrelation+= (i, k);
20:    end if
21:    V_i = V_k;   \* The next node is now V_i *\
22:  end while
23: end for
```

nodes. And cross correlation properties are always defined in pairs. Therefore, all combination of two paths from the path vector $\bar{P}_{ij}$ is considered. If there are N such paths in the vector, there are $\frac{N.(N-1)}{2}$ number of paths.

Let *GetPathNodes(p_{ij}^k)* be a function which returns all the nodes including V_i and V_j along the k^{th} path between them. Consider any pair (k,l) of paths from the vector $\bar{P}_{k,l}$ and let $\bar{V}_{k,l}$ be the list of nodes common to both paths. Apart from the nodes V_i and V_j, there can be more than one node in this list. In such a case, the paths re-converge more than once.

Between any two paths (k,l) considered in pairs, it has to be checked if there are common noise sources to the multi-output sub-system V_i. This is known by checking if any of the operator noise sources have a path to both these outputs. If this correlation does not exist, the two noise source in spite of them being from the same sub-system are uncorrelated and hence nothing else needs to be done in this case and the loop is iterated to consider the next pair of paths. Otherwise, the two outputs of the sub-system V_i are correlated with one another. Therefore, the flag *SrcCorr* is set and the *OutputXCorr* list is appended with the pair (k,l).

Consider any intermediate node $V_{k,l}$ other than V_i and V_j and that the *SrcCorr* flag

Procedure A.3 : AnnotateXCorrParams($\bar{P}_{ij}, V_i, V_j, H(V,E)$)

```
1:  \* Consider various paths in P̄_ij in pairs *\
2:  for all Pairs : (p^k_ij, p^l_ij) such that p^k_ij, p^l_ij ∈ P̄_ij do
3:    \* Check for cross-correlated output pairs of node V_i *\
4:    C_k(ν, e) = GetOutputCone(k, V_i);
5:    C_l(ν, e) = GetOutputCone(l, V_i);
6:    \* Check for common operators shared between cones (k, l) of node V_i *\
7:    if {ν_k ∈ C_k} ∩ {ν_l ∈ C_l} ≠ {0} then
8:      V_i.OutputXCorr += (k, l);  \* add pair (k, l) to cross-correlation list *\
9:    else
10:     continue;  \* No common nodes. Hence, no cross correlation *\
11:   end if
12:   \* Check for correlation with input signals *\
13:   \* Get the nodes common in both paths *\
14:   V̄_k,l = GetPathNodes(p^k_ij) ⋂ GetPathNodes(p^l_ij);
15:   for all nodes: V_k,l ∈ V̄_k,l and V_k,l ≠ V_i and V_k,l ≠ V_j do
16:     V_k = GetPrevNode(p^k_ij, V_k,l);
17:     V_l = GetPrevNode(p^k_ij, V_k,l);
18:     V_m = GetNextNode(p^k_ij, V_k,l);
19:     V_n = GetNextNode(p^l_ij, V_k,l);
20:       \* Require the input correlation between inputs (k, l) *\
21:     V_k,l.InputXCorr += (k, l);
22:       \* If there is divergence again, mark output cross-correlation *\
23:     if m ≠ n then
24:       V_k,l.OutputXCorr += (m, n);
25:     end if
26:       \* Proceed backward from V_k,l along p^k_ij until it reaches V_i. *\
27:     while V_k ≠ V_i do
28:       V_k̀ = GetPrevNode(p^k_ij, V_k);
29:         \* Mark for input-output cross correlation *\
30:       V_k.TransXCorr += (k, k̀);
31:       V_k = V_k̀;
32:     end while
33:       \* Proceed backward from V_k,l along p^l_ij until it reaches V_i. *\
34:     while V_l ≠ V_i do
35:       V_l̀ = GetPrevNode(p^l_ij, V_l);
36:         \* Mark for input-output cross correlation *\
37:       V_l.TransXCorr += (l, l̀);
38:       V_l = V_l̀;
39:     end while
40:   end for
41: end for
```

of the node $V_{k,l}$ is set. It is clear that the input edges of the node $V_{k,l}$ along the k^{th} and l^{th} paths have emanated from the same source. Therefore, the corresponding inputs are correlated. So, the *InputXCorr* list of the node $V_{k,l}$ is appended with (k, l). Here, (k, l) are the indices of the nodes occurring prior to the node $V_{k,l}$ along the paths k and l respectively. If the paths k and l re-converge more than once the outputs of the node $V_{k,l}$ along the two paths considered are correlated and hence the list *OutputXCorr* is also appended with (m, n). Here, m, n are the indices of the nodes succeeding the node $V_{k,l}$ along the paths k and l respectively.

The k^{th} and the l^{th} output of node V_i is correlated. To propagate this information to the input of the node $V_{k,l}$, it is necessary to keep the cross correlation information between the input and the output along the paths k and l. This is done by tracing the path back from node $V_{k,l}$ to node V_i. All the successive indices $\grave{k}$ and $\grave{l}$ occurring prior to $V_{k,l}$ along paths k and l are appended in the *TransXCorr* list of the nodes V_k and V_l respectively.

References

[1] A. Ahmadi and M. Zwolinski. Word-length oriented multiobjective optimization of area and power consumption in dsp algorithm implementation. In *25th International Conference on Microelectronics, 2006*, pages 614 –617, 0-0 2006. 34

[2] T. Arslan and D.H. Horrocks. A genetic algorithm for the design of finite word length arbitrary response cascaded iir digital filters. In *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*, pages 276 –281, sep 1995. 141

[3] G. Baicher. Optimization of finite word length coefficient iir digital filters through genetic algorithms a comparative study. In Licheng Jiao, Lipo Wang, Xinbo Gao, Jing Liu, and Feng Wu, editors, *Advances in Natural Computation*, **4222** of *Lecture Notes in Computer Science*, pages 641–650. Springer Berlin / Heidelberg, 2006. 141

[4] A. Banciu, E. Casseau, D. Menard, and T. Michel. Stochastic modeling for floating-point to fixed-point conversion. *IEEE Workshop on Signal Processing Systemsm, SIPS2011*, pages 180–185, 2011. 19, 102

[5] P.H. Bauer and L.-J. Leclerc. A computer-aided test for the absence of limit cycles in fixed-point digital filters. *IEEE Transactions on Signal Processing*, **39**[11]:2400 –2410, nov 1991. 29

[6] P. Belanovic and M. Lesser. A library of parameterized floating-point modules and their use. *International Conference on Field Programmable Logic and Applications, FPL2002*, **2002**:657–666, 2002. 17

[7] F. Berens and N. Naser. Algorithm to system-on-chip design flow that leverages system-studio and systemc. *Synopsys Inc.*, 2004. 22

[8] Bluespec. Bluespec compiler. *http://www.bluespec.com/*. 3

[9] D. Boland and G.A. Constantinides. Bounding variable values and round-off effects using handelman representations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, **30**[11]:1691 –1704, nov. 2011. 28

[10] T. Bose and M.-Q. Chen. Overflow oscillations in state-space digital filters. *Circuits and Systems, IEEE Transactions on*, **38**[7]:807 –810, jul 1991. 29

[11] T. Bose and M.-Q. Chen. Stability of digital filters implemented with two's complement truncation quantization. *IEEE Transactions on Signal Processing*, **40**[1]:24 –31, jan 1992. 29

[12] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004. 178

[13] G. Caffarena, C. Carreras, J. A. López, and Á. Fernández. Sqnr estimation of fixed-point dsp algorithms. *EURASIP Journal on Advanced Signal Processing*, **2010**:21:1–21:12, February 2010. 27

[14] G Caffarena, J. A. López, G. Leyva, C. Carreras, and O. Nieto-Taladriz. Architectural synthesis of fixed-point dsp datapaths using fpgas. *International Journal on Reconfigurable Computing*, **2009**:8:1–8:14, January 2009. 21

[15] Gabriel Caffarena, Juan A. López, Gerardo Leyva, Carlos Carreras, and Octavio Nieto-Taladriz. Architectural synthesis of fixed-point dsp datapaths using fpgas. *Int. J. Reconfig. Comput.*, **2009**:8:1–8:14, jan 2009. 29

[16] J.D.O. Campo, F. Cruz-Roldan, and M. Utrilla-Manso. Tighter limit cycle bounds for digital filters. *Signal Processing Letters, IEEE*, **13**[3]:149 – 152, march 2006. 29

[17] M.-A. Cantin, Y. Savaria, and P. Lavoie. A comparison of automatic word length optimization procedures. In *IEEE International Symposium on Circuits and Systems, ISCAS 2002*, **2**, pages II–612 – II–615 vol.2, 2002. 33

[18] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie. An automatic word length determination method. In *The 2001 IEEE International Symposium on Circuits and Systems, ISCAS 2001*, **5**, pages 53 –56 vol. 5, 2001. 33, 141, 151

[19] C. Carreras, J.A. Lopez, and O. Nieto-Taladriz. Bit-width selection for data-path implementations. In *System Synthesis, 1999. Proceedings. 12th International Symposium on*, pages 114 –119, nov 1999. 19, 28

[20] F. Catthoor, H. De Man, and J. Vandewalle. Simulated-annealing-based optimization of coefficient and data word-lengths in digital filters. *International Journal of Circuit Theory and Applications*, **16**[4]:371–390, 1988. 34

[21] S.C. Chan and K.M. Tsui. Wordlength determination algorithms for hardware implementation of linear time invariant systems with prescribed output accuracy. In *IEEE International Symposium on Circuits and Systems, ISCAS 2005*, pages 2607 – 2610 Vol. 3, may 2005. 35, 141, 172

[22] S.C. Chan and K.M. Tsui. Wordlength optimization of linear time-invariant systems with multiple outputs using geometric programming. *IEEE Transactions on Circuits and Systems I: Regular Papers*, **54**[4]:845 –854, april 2007. 35

[23] H. Choi and W.P. Burleson. Search-based wordlength optimization for vlsi/dsp synthesis. In *VII Workshop on VLSI Signal Processing 1994*, pages 198 –207, 1994. 32, 34

[24] A. Cilio and H. Corporaal. Floating point to fixed point conversion of c code. *Lecture Notes in Computer Science*, **1575**:1–99, 1999. 36

[25] T. Claasen, W. Mecklenbrauker, and J. Peek. Effects of quantization and overflow in recursive digital filters. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, **24**[6]:517 – 529, dec 1976. 29

[26] M. Clark, M. Mulligan, D. Jackson, and D. Linebarger. Accelerating Fixed-Point Design for MB-OFDM UWB Systems. *CommsDesign*, January 2005. 4

[27] J.A. Clarke, G. A. Constantinides, and P. Y. K. Cheung. Word-length selection for power minimization via nonlinear optimization. *ACM Transactions on Design Automation in Electronic Systems*, **14**[3]:39:1–39:28, June 2009. 32, 36

[28] R. Cmar, L. Rijnders, P. Schaumont, S. Vernalde, and I. Bolsens. A methodology and design environment for dsp asic fixed point refinement. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '99, New York, NY, USA, 1999. ACM. 37

[29] J. Cong, K. Gururaj, B. Liu, C. Liu, Z. Zhang, S. Zhou, and Yi Zou. Evaluation of static analysis techniques for fixed-point precision optimization. *Proceedings of 17th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM2009*, pages 231–234, 2009. 19

[30] G. A. Constantinides. Perturbation Analysis for Word-length Optimization. In *IEEE Symposium on Field-Programmable Custom Computing Machines (FFCM'03)*, 2003. 26

[31] G. A. Constantinides, P. Y. K. Cheung, and W. Luk. Wordlength optimization for digital signal processing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **22**[10]:1432–1442, 2003. 20, 31, 32, 33

[32] G. A. Constantinides, P. Y. K. Cheung, and L. Wayne. *Synthesis and Optimization of DSP Algorithms*. Springer, 1^{st} edition, 2004. 39

[33] G. A. Constantinides, Peter Y. K. Cheung, and Wayne Luk. Multiple precision for resource minimization. In *Proceedings of the 2000 IEEE Symposium on Field-Programmable Custom Computing Machines*, FCCM '00, pages 307–, Washington, DC, USA, 2000. IEEE Computer Society. 37

[34] G. A. Constantinides and G. Woeginger. The complexity of multiple wordlength assignment. *Applied Mathematics Letters*, **15(2)**:137–140, 2002. 3, 32, 141

[35] CoWare. Coware spw. *Technical Report, CoWare*, 2010. 21

[36] Novo Bruna David. Exploiting adaptive precision in software defined radios. *Doctoral dissertation, K. U. Leuven, IMEC, Belgium*, 2010. 39

[37] L. De Coster, M. Ade, R. Lauwereins, and J. Peperstraete. Code generation for compiled bit-true simulation of dsp applications. *Proceedings of the 11th International Symposium on System Synthesis 1998*, pages 9 –14, dec 1998. 22

[38] F. de Dinechin, C. Klein, and B. Pasca. Generating high-performance custom floating-point pipelines. *International Conference on Field Programmable Logic and Applications, FPL2009*, **2009**:59–64, 2009. 17

[39] J. Eker, J.W. Janneck, E.A. Lee, Jie Liu, Xiaojun Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Yuhong Xiong. Taming heterogeneity - the ptolemy approach. *Proceedings of the IEEE*, **91**[1]:127 – 144, jan 2003. 21

[40] Impulse *accelerated technologies*. Impulse C. *http://www.impulseaccelerated.com/*, 2012. 3

[41] C. Ewe, P. Cheung, and G. A. Constantinides. Error modeling of dual fixed-point arithmetic and its application in field programmable logic. *International Conference on Field Programmable Logic and Applications, FPL2005*, **2005**:124–129, 2005. 18

[42] Claire Fang Fang, Rob A. Rutenbar, Markus Püschel, and Tsuhan Chen. Toward efficient static analysis of finite-precision effects in dsp applications via affine arithmetic modeling. *Proceedings of the 40th annual Design Automation Conference, DAC2003*, pages 496–501, 2003. 19

[43] P.D. Fiore. Efficient approximate wordlength optimization. *IEEE Transactions on Computers*, **57**[11]:1561 –1570, nov. 2008. 35

[44] P.D. Fiore and L. Lee. Closed-form and real-time wordlength adaptation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing 1999*, **4**, pages 1897 –1900 vol.4, mar 1999. 35

[45] A. Gaffar, O. Mencer, W. Luk, P. Cheung, and N. Shirazi. Floating-point bit-width analysis via automatic differentiation. *International Conference on Field Programmable Logic and Applications, FPL2002*, **2002**:158–165, 2002. 18

[46] GCC. GNU C Compiler. *http://www.gnu.org/*. 3

[47] GeCoS. Generic compiler suite. *http://gecos.gforge.inria.fr/*. 38

[48] M. C. Grant and S. P. Stephen P. Boyd Boyd. *The CVX Users Guide*. CVX Research, Inc., Cambridge, UK, 2012. 190

[49] K. Han. Automating transformations from oating-point to xedpoint for implementing digital signal processing algorithms. In *Ph.D. dissertation, University of Texas at Austin,*, 2006. 35

[50] K. Han and B.L. Evans. Optimum wordlength search using sensitivity information. *EURASIP Journal on Applied Signal Processing*, **2006**:76–76, January 2006. 32, 33

[51] Kyungtae Han, Iksu Eo, Kyungsu Kim, and Hanjin Cho. Numerical word-length optimization for cdma demodulator. In *The 2001 IEEE International Symposium on Circuits and Systems, ISCAS 2001.*, **4**, pages 290 –293 vol. 4, may 2001. 33

[52] C. He, G. Qin, M. Lu, and W. Zhao. An efficient implementation of high-accuracy finite difference computing engine on fpgas. *International Conference on Application Specific Systems, Architectures and Processors*, **2006**:95–98, 2006. 18

[53] Shousheng He and M. Torkelson. Designing pipeline fft processor for ofdm (de)modulation. In *Signals, Systems, and Electronics, 1998. ISSSE 98. 1998 URSI International Symposium on*, pages 257 –262, sep-2 oct 1998. 77

[54] Nicolas Herve. Contributions la synthse d'architecture virgule fixe largeurs multiples. *Doctoral thesis, University of Rennes-1, E.N.S.S.A.T., Lannion*, 2007. 175

[55] T. Hinamoto, S. Yokoyama, T. Inoue, W. Zeng, and Wu-Sheng Lu. Analysis and minimization of l2-sensitivity for linear systems and two-dimensional state-space filters using general controllability and observability gramians. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **49**[9]:1279 – 1289, sep 2002. 28

[56] C. H. Ho, C. W. Yu, P. Leong, W. Luk, and S. Wilton. Floating-point fpga: Architecture and modeling. *IEEE. Transactions on Very Large Scale Integration Systems*, **17**:1709–1718, 2009. 17

[57] Apple Inc. Apple macbook pro, version 10.5.8. 2008. 168, 170, 171, 194

[58] L.B. Jackson. On the interaction of roundoff noise and dynamic range in digital filters. *The Bell System Technical Journal*, **49**[2], Feb. 1970. 19

[59] M. Jersak and M. Willems. Fixed-point extended c compiler allows more efficient high-level programming of fixed-point dsps. In *Proceedings of the International Conference on Signal Processing Applications and Technology ICSPAT 1998*, pages 1–5, 1998. 37

[60] G.E. Johnson. Constructions of particular random processes. *Proceedings of the IEEE*, **82**[2]:270 –285, feb 1994. 50, 82

[61] K. Kalliojarvi and J. Astola. Roundoff errors in block-floating-point systems. *Signal Processing, IEEE Transactions on*, **44**[4]:783 –790, apr 1996. 18

[62] J. Kauraniemi. Analysis of limit cycles in the direct form delta operator structure by computer-aided test. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 1997*, **3**, pages 2177 –2180 vol.3, apr 1997. 29

[63] H. Keding. Pain killers for fixed-point design flow. *Technical Report, Synopsys*, 2010. 21

[64] H. Keding, M. Coors, O. Lüthje, and H. Meyr. Fast bit-true simulation. *Proceedings of the 38th annual Design Automation Conference*, pages 708–713, 2001. 22

[65] J. F. Kenny and E. S. Keeping. Cumulants and the cumulant-generating function, additive property of cumulants, and sheppards correction. In *Mathematics of Statistics, 2nd ed. Princeton, NJ: Van Nostrand*, **2**, pages 77–82, 1951. 69

[66] S. Kim, Ki-Il. Kum, and W. Sung. Fixed-point optimization utility for C and C++ based digital signal processing programs. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, **45**:1455–1464, 1998. 19, 21, 22

[67] S. Kim and W. Sung. Fixed-point-Simulation Utility for C and C++BasedDigitalSignalProcessingPrograms. *Record of the Twenty-Eighth Asilomar Conference on Signals, Systems and Computers, 1994*, **1**:162–166, 1994. 36

[68] S. Kim and W. Sung. A floating-point to fixed-point assembly program translator for the TMS320C25. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, **41**[11]:730–739, 1994. 19, 36

[69] R. R. Kinnison. *Applied Extreme Value Statistics*. Macmillian Publishing Company, 1985. 20

[70] K. I. Kum and W. Sung. Combined word-length optimization and high-level synthesis of digital signal processing systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **20**[8]:921–930, 2001. 20

[71] D.-U. Lee, A.A. Gaffar, R.C.C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides. Accuracy-guaranteed bit-width optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **25**[10]:1990 –2000, oct. 2006. 28

[72] D.-U. Lee, A.A. Gaffar, R.C.C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides. Accuracy-guaranteed bit-width optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **25**[10]:1990 –2000, oct. 2006. 34

[73] D-U. Lee, A.A. Gaffar, O. Mencer, and W. Luk. Minibit: bit-width optimization via affine arithmetic. In *Proceedings of the 42nd annual Design Automation Conference DAC 2005*, DAC '05, pages 837–840, New York, NY, USA, 2005. ACM. 28

[74] J. Lee, R. Haralick, and L. Shapiro. Morphologic edge detection. *Robotics and Automation, IEEE Journal of*, **3**[2]:142 –156, april 1987. 128

[75] G. Li, M. Gevers, and Youxian S. Performance analysis of a new structure for digital filter implementation. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, **47**[4]:474 –482, apr 2000. 28

[76] Min Li, B. Bougard, E.E. Lopez, A. Bourdoux, D. Novo, L. Van Der Perre, and F. Catthoor. Selective spanning with fast enumeration: A near maximum-likelihood mimo detector designed for parallel programmable baseband architectures. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 737 –741, may 2008. 128, 135

[77] J. A. López, G. Caffarena, and C. Carreras. Fast and accurate computation of l2 sensitivity in digital filter realizations. In *Technical Report, University Politechnica de Madrid*, 2006. 29

[78] J.A. Lopez, G. Caffarena, C. Carreras, and O. Nieto-Taladriz. Analysis of limit cycles by means of affine arithmetic computer-aided tests. In *12th European Signal Processing Conference (EUSIPCO 2004)*, pages 991–994, 2004. 29

[79] J.A. Lopez, G. Caffarena, C. Carreras, and O. Nieto-Taladriz. Fast and accurate computation of the roundoff noise of linear time-invariant systems. *IET Circuits, Devices Systems*, **2**[4]:393 –408, aug. 2008. 27

[80] J.A. Lopez, C. Carreras, G. Caffarena, and O. Nieto-Taladriz. Fast characterization of the noise bounds derived from coefficient and signal quantization. In *Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on*, **4**, pages IV–309 – IV–312 vol.4, may 2003. 28

[81] J.A. Lopez, C. Carreras, and O. Nieto-Taladriz. Improved interval-based characterization of fixed-point lti systems with feedback loops. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, **26**[11]:1923 –1933, nov. 2007. 19, 28

[82] Mathworks. Fixed-point blockset user's guide (ver. 2.0). 2001. 21

[83] P.K. Meher, J. Valls, Tso-Bing Juang, K. Sridharan, and K. Maharatna. 50 years of cordic: Algorithms, architectures, and applications. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, **56**[9]:1893 –1907, sept. 2009. 161

[84] G. Melquiond. Gappa. *http://gappa.gforge.inria.fr/ version 0.16.1*, 2012. 19

[85] D. Menard. Id.fix - infrastructure for the design of fixed-point systems. In *University Booth of the IEEE/ACM Conference on Design, Automation and Test in Europe (DATE)*, 2011. 38

[86] D. Menard, D. Chillet, and O. Sentieys. Floating-to-fixed-point Conversion for Digital Signal Processors. *EURASIP Journal on Applied Signal Processing, Special issue on Design Methods for DSP Systems*, **2006**:1–19, january 2006. 34

[87] D. Menard, R. Rocher, and O. Sentieys. Analytical fixed-point accuracy evaluation in linear time-invariant systems. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, **55**[10]:3197 –3208, nov. 2008. 26, 28, 38, 68

[88] D. Menard and O. Sentieys. Automatic evaluation of the accuracy of fixed-point algorithms. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, 2002.*, pages 529 –535, 2002. 27

[89] W. Mills, C. Mullis, and R. Roberts. Digital filter realizations without overflow oscillations. *IEEE Transactions on Acoustics, Speech and Signal Processing*, **26**[4]:334 – 338, aug 1978. 29

[90] S.K. Mitra. *Digital Signal Processing Laboratory Using MATLAB*. WCB/McGrawHill, University of California, Santa Barbara, 1999. 29, 159

[91] R. E. Moore. *Inerval Analysis*. Prentice-Hall, 1966. 18

[92] R. Nambiar, C.K.K. Tang, and P. Mars. Genetic and learning automata algorithms for adaptive digital filters. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 1992*, **4**, pages 41 –44 vol.4, mar 1992. 34

[93] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee. Precision and error analysis of matlab applications during automated hardware synthesis for fpgas. In *Proceedings of the conference on Design, automation and test in Europe*, DATE '01, pages 722–728, Piscataway, NJ, USA, 2001. IEEE Press. 37

[94] M. Nemani and F.N. Najm. High-level area and power estimation for vlsi circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **18**[6]:697 –713, jun 1999. 29

[95] S.C. Ng, S.H. Leung, C.Y. Chung, A. Luk, and W.H. Lau. The genetic search approach. a new learning algorithm for adaptive iir filtering. *IEEE Signal Processing Magazine*, **13**[6]:38 –46, nov 1996. 34

[96] H.-N. Nguyen, D. Menard, and O. Sentieys. Novel algorithms for word-length optimization. In *19th European Signal Processing Conference (EUSIPCO 2011)*, pages 1944–1948, 2011. 33, 35

[97] Z. Ning, B. Haller, and R. Brodersen. Systematic architecture exploration for implementing interference suppression techniques in wireless receivers. In *IEEE Workshop on Signal Processing Systems, SiPS 2000*, pages 218 –227, 2000. 29

[98] M. Novey, T. Adali, and A. Roy. A complex generalized gaussian distribution characterization, generation, and estimation. *Signal Processing, IEEE Transactions on*, **58**[3]:1427 –1433, March 2010. 71

[99] D. Novo, A. Kritikakou, P. Raghavan, L. Van der Perre, J. Huisken, and F. Catthoor. Ultra low energy domain specific instruction-set processor for on-line surveillance. *IEEE 8th Symposium on Application Specific Processors (SASP)*, **2010**:30–35, 2010. 17

[100] A. Oppenheim. Realization of digital filters using block-floating-point arithmetic. *Audio and Electroacoustics, IEEE Transactions on*, **18**[2]:130 – 136, jun 1970. 18

[101] Emre Özer, Andy P. Nisbet, and David Gregg. A stochastic bitwidth estimation technique for compact and low-power custom processors. *ACM Transactions on Embedded Computer Systems*, **7**:34:1–34:30, 2008. 19

[102] Emre Özer, Andy P. Nisbet, and David Gregg. A stochastic bitwidth estimation technique for compact and low-power custom processors. *ACM Transactions on Embedded Computer Systems*, **7**:34:1–34:30, 2008. 20

[103] S. Papoulis and S. Unnikrishna Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill Europe, 4^{th} edition, 2002. 113

[104] Keshab K. Parhi. *VLSI Digital Signal Processing Systems: Design and Implementation*. Wiley-Interscience, 1^{st} edition, 1999. 118

[105] Tamar Peli and David Malah. A study of edge detection algorithms. *Computer Graphics and Image Processing*, **20**[1]:1–21, 1982. 131

[106] K. Peppas, F. Lazarakis, D. Axiotis, T. Al-Gizawi, and A. Alexandridis. System level performance evaluation of mimo and siso ofdm-based wlans. *Wireless Networks*, **15**:859–873, 2009. 135

[107] K. Premaratne, E.C. Kulasekere, P.H. Bauer, and L.J. Leclerc. An exhaustive search algorithm for checking limit cycle behavior of digital filters. In *IEEE International Symposium on Circuits and Systems, ISCAS 1995*, **3**, pages 2035 –2038 vol.3, apr-3 may 1995. 29

[108] R. Rocher, D. Menard, P. Scalart, and O. Sentieys. Analytical accuracy evaluation of fixed-point systems. In *15th European Signal Processing Conference (EUSIPCO 2007)*, pages 999–1003, 2007. 28, 38

[109] Romuald Rocher, Daniel Menard, Nicolas Herve, and Olivier Sentieys. Fixed-point configurable hardware components. *EURASIP Journal on Embedded Systems*, **2006**[1]:1–13, January 2006. 21

[110] A. Savich, M. Moussa, and S.Areibi. The impact of arithmetic representation on implementing mlp-bp on fpgas: A study. *IEEE. Transactions on Neural Networks*, **8**:240–252, 2007. 18

[111] C. Shi and R.W. Brodersen. Automated fixed-point data-type optimization tool for signal processing and communication systems. In *Proceedings of the 41st Design Automation Conference, DAC2004.*, pages 478 –483, july 2004. 29, 37

[112] C. Shi and R.W. Brodersen. A perturbation theory on statistical quantization effects in fixed-point dsp with non-stationary inputs. In *Proceedings of the 2004 International Symposium on Circuits and Systems, ISCAS 2004*, **3**, pages III – 373–6 Vol.3, may 2004. 28

[113] Frank Y. Shih. *Image Processing and Mathematical Morphology: Fundamentals and Applications*. CRC Press, 1^{st} edition, 2009. 131

[114] N. Sulaiman. A multi-objective genetic algorithm for on-chip real-time optimisation of word length and power consumption in a pipelined fft processor targeting a mc-cdma receiver. In *Proceedings of the 2005 NASA/DoD Conference on Evolvable Hardware*, EH '05, pages 154–159, Washington, DC, USA, 2005. IEEE Computer Society. 34

[115] W. Sung and Kum. Ki-Il. Word-length determination and scaling software for a signal flow block diagram. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, **ii**, pages II/457 –II/460 vol.2, apr 1994. 32

[116] W. Sung and Ki-Il Kum. Simulation-based word-length optimization method for fixed-point digital signal processing systems. *IEEE Transactions on Signal Processing*, **43**[12]:3087 –3090, dec 1995. 33

[117] Synopsys. Synopsys prime time suite. *http://www.synopsys.com*. 175

[118] P. Vaidyanathan and V. Liu. An improved sufficient condition for absence of limit cycles in digital filters. *IEEE Transactions on Circuits and Systems*, **34**[3]:319 – 322, mar 1987. 29

[119] J.-W. Weijers, V. Derudder, S. Janssens, F. Petré, and A. Bourdoux. From mimo-ofdm algorithms to a real-time wireless prototype: a systematic matlab-to-hardware design flow. *EURASIP J. Appl. Signal Process.*, **2006**:138–138, January 2006. 38

[120] B. Widrow and I. Kollár. *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge University Press, Cambridge, UK, 2008. 23, 24, 86, 94, 100

[121] M. Willems, V. Bursgens, T. Grotker, and H. Meyr. Fridge: An interactive code generation environment for hw/sw codesign. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-97*, **1**, pages 287 –290 vol.1, apr 1997. 28, 36

[122] P.W. Wolniansky, G.J. Foschini, G.D. Golden, and R.A. Valenzuela. V-blast: an architecture for realizing very high data rates over the rich-scattering wireless channel. In *Signals, Systems, and Electronics, 1998. ISSSE 98. 1998 URSI International Symposium on*, pages 295 –300, sep-2 oct 1998. 71

[123] Xilinx. System generator for dsp. *http://www.xilinx.com*. 22

[124] C. W. Yu, J. Lamoureux, S. Wilton, P. Leong, and W. Luk. The coarse-grained/ fine-grained logic interface in fpgas with embedded floating-point arithmetic. *International Journal of Reconfigurable Computing*, **2008**:1–10, 2008. 17

VU :

Le Directeur de Thèse

Olivier Sentieys

VU :

Le Responsable de l'École Doctorale

VU pour autorisation de soutenance

Rennes, le

Le Président de l'Université de Rennes 1

Guy CATHELINEAU

VU après autorisation pour autorisation de publication :

Le Président de Jury,