Accelerated Performance Evaluation of Fixed-Point Systems With Un-Smooth Operations

Karthick Nagaraj Parashar, Daniel Menard, *Member, IEEE,* and Olivier Sentieys, *Member, IEEE*

***Abstract*—The problem of accuracy evaluation is one of the most time consuming tasks during the fixed-point refinement process. Analytical techniques based on perturbation theory have been proposed in order to overcome the need for long fixed-point simulation. However, these techniques are not applicable in the presence of certain operations classified as un-smooth operations. In such circumstances, fixed-point simulation should be used. In this paper, an algorithm detailing the hybrid technique which makes use of an analytical accuracy evaluation technique used to accelerate fixed-point simulation is presented. This technique is applicable to signal processing systems with both feed-forward and feedback interconnect topology between its operations. The acceleration obtained as a result of applications of the proposed technique is consistent with fixed-point simulation, while reducing the time taken for fixed-point simulation by several orders of magnitude.**

***Index Terms*—Accuracy estimation, fixed-point refinement, perturbation theory, quantization noise, signal processing.**

I. Introduction

In the era of digital convergence, a number of functionalities such as communication, security, and entertainment are brought together on a single electronic device. From a designer's perspective, these new-generation devices seek higher computational power, while at the same time keeping the costs lower. The semiconductor technology scaling over the past couple of decades has made the devices more energy efficient, more compact, and much faster resulting in an exponential increase in the computational performance per unit energy, area, and time. As the technology scaling is hitting a point of saturation, the phenomena of achieving higher performance at lower costs by simply scaling the technological parameters are approaching saturation.

The use of fixed-point arithmetic for performing computations comes with several benefits, such as increase in the speed of execution, reduced area, and low-energy foot-print. However, the downside of this approach is that the computations will have to be carried out with lower precision. The impact of reduced precision may or may not be acceptable, depending on the application or the algorithm for which they are used. Signal processing algorithms form an interesting class of applications which can indeed be functionally correct in spite of reduced precision. Of course, each algorithm has a degree of severity in loss of its precision. Making the choice of fixed-point word-lengths optimally in order to strike a good balance between the performance parameters and accuracy is popularly referred to as the word-length optimization (WLO) problem. The process of taking an algorithm through such an optimization procedure is usually referred to as fixed-point refinement of the algorithm.

Consider a system with M signals whose word-lengths (in number of bits) are represented by the vector $\mathbf{w} = [w_1, w_2, \dots w_M]$. If the loss in accuracy due to the assigned fixed-point word-lengths is given as the function $\lambda(\mathbf{w})$, the cost as a function $C(\mathbf{w})$ and the maximum loss in computational accuracy is λ_{obj}, the WLO problem is written as

$$\min\left(C\left(\mathbf{w}\right)\right) \quad \text{subject to} \quad \lambda\left(\mathbf{w}\right) \leq \lambda_{obj}. \tag{1}$$

The WLO problem is known to be NP-hard [1]. A number of heuristic-based optimization techniques have been proposed in order to solve this problem. These heuristics are iterative in nature, and they require repeated evaluation of the loss in accuracy and implementation cost considerations for different cases of assigned word-lengths. The complexity of some popular heuristics such as the min +1 bit algorithm [2] tends to grow exponentially with the number of optimization variables. This means that the number of times it is required to evaluate functions $\lambda(\mathbf{w})$ and $C(\mathbf{w})$ grows exponentially with increasing number of optimization variables.

In a typical high-level synthesis (HLS) flow, the actual cost of the implementation is influenced by a number of decision steps during RTL generation and synthesis such as scheduling, binding to resources. Since the fixed-point refinement step is performed very early (much before RTL synthesis), it is difficult to make accurate estimates. A first cut estimate of the cost using either the area or an estimate of energy consumption that increases with the number of bits assigned to fixed-point word-lengths of fixed-point operations is used. Although this approximation does not give the accurate value of the cost, it provides a reliable, but coarse, trend while simplifying the process of estimation.

The loss in computational accuracy is measurable at the early stage of fixed-point assignment and its effect can be estimated accurately by considering the functionality and the data-path of the given signal processing system. Several techniques

Manuscript received May 21, 2013; revised October 7, 2013; accepted October 29, 2013. Date of current version March 17, 2014. This work was supported in part by the European Union under the Seventh Framework Program under Grant ICT-287733 and in part by the ANR DEFIS Project. This paper was recommended by Associate Editor L. Pozzi.

K. N. Parashar is with the Circuits and Systems Group, Imperial College, London SW7 2AZ, U.K. (e-mail: karthick.np@gmail.com).

D. Menard is with the Department of Electrical and Computer Engineering, Engineering School, Institut National des Sciences Appliquées de Rennes, Rennes 35708, France (e-mail: daniel.menard@insa-rennes.fr).

O. Sentieys is with Equipe CAIRN, INRIA, Lannion 22300, France (e-mail: olivier.sentieys@irisa.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2013.2292510

based on both simulation and analytical techniques have been proposed in the past. Simulation-based approaches focus on creating infrastructures for emulation of fixed-point data types using custom libraries [3]. Such libraries allow specification of fixed-point formats with arbitrary lengths for the purpose of experimentation. While these approaches are useful and have contributed significantly to the development of several fixed-point design tools [4], [5], the time required for simulation can be very long. Time required for optimization grows with the number of operations in the system (system size), and the results obtained are specific to the input data set provided. As in any other simulation, the onus of providing input data sets which is truly representative of all possible scenarios is on the user of such a tool. In the case of large systems, both the size of input data set and the fixed-point search can be very large, and hence, performance evaluation by simulation proves difficult to be useful practically. Analytical techniques [6]–[8] for error estimation make use of well established stochastic models for errors due to finite precision [9]. Such techniques incur an initial overhead of evaluating the closed-form expression $\lambda(\mathbf{w})$ specific for a given system or signal processing algorithm. Automatically doing this requires a fair degree of semantic analysis and also capturing some floating-point data-points to for reference. Once this expression is obtained, the time taken for its evaluation is insignificant and it can be repeated for any given word-length assignment. However, the application of such analytical techniques is limited to systems with certain types of operations referred to as smooth. In the presence of un-smooth operations, the stochastic model fails to capture the error behavior due to fixed-point numbers, and it becomes inevitable to resort to simulation-based techniques. In [10] and [11], analytical techniques for estimating the impact of errors due to quantization in the presence of un-smooth operations are explored. The complexity of these techniques is very high, and at the same time, these techniques are not applicable in the presence of feedback loops in the signal flow graph.

In this paper, the problem of fixed-point performance evaluation of systems in the presence of un-smooth operations is considered. A hybrid approach that makes use of partial analytical results for accelerating the fixed-point simulation algorithm is proposed. This approach, to the best of our knowledge, is the first of its kind to provide an alternative to classical fixed-point simulation in the presence of un-smooth operations. The proposed method promises acceleration of fixed-point simulation of any given signal processing algorithm without imposing any topological constraints. The rest of this paper is organized as follows. The next section is dedicated to classification of operations into smooth and un-smooth. In Section III the hybrid algorithm for accelerating fixed-point simulation is presented and discussed. This algorithm makes use of the definitions provided in Section II. Results showing acceleration in fixed-point simulation obtained due to the application of the hybrid method are described in Section IV. This paper is summarized and concluded in Section V.

II. Un-Smooth Operations

The behavior of signal processing systems implemented using fixed-point representation of signals suffers due to the limited dynamic range and precision of fixed-point number format. The fixed-point representation of signals manifests as a number of error sources in the system. The errors introduced into the system due to the dynamic range limitation are either due to saturation or overflow effects. These errors can be contained or eliminated altogether by providing for a sufficient number of bits in the integer part. The error introduced due to limited precision is referred to as the quantization error. The stochastic behavior of quantization errors is completely characterized by Widrow's quantization noise model [pseudo-quantization-noise (PQN) [9]].

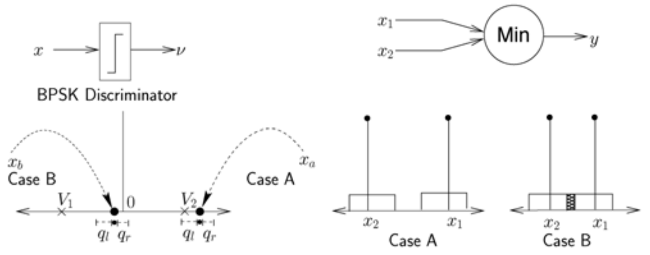

Fig. 1. Propagation of quantization errors through un-smooth operations.

The error generated due to quantization propagates through the datapath in the system to its output. The datapath consists of a number of operations, and the value of quantization noise at the output depends on the functionality of the path it takes to the system output. In [12], a technique based on the perturbation theory for propagation of errors due to quantization through any given n-ary[1] operation is defined. In this approach, the error values at the output of the operation are approximated by the first-order Taylor series expansion of the function describing the operation. A limitation of this approach is that the function describing the operation has to be continuous and at least first-order differentiable with respect to each of its n inputs. Therefore, an n-ary operation that is first order differentiable with respect to all its n inputs is considered smooth. In the presence of smooth operations, the propagation of quantization errors can be determined using analytical techniques described in [12] and [13].

Definition: The operation whose response to perturbation in one of its input values is a proportional perturbation in its output value is defined as smooth operation. An un-smooth operation can, therefore, be defined as an operation that is not smooth.

In a fixed-point implementation, the source for these perturbations is the error due to finite precision operations. If the quantization step-size is small in comparison to the signal, errors due to quantization are small in magnitude and qualify to be classified as small perturbations. The result of perturbing signals at the input of an un-smooth operation can be nonlinear and can have and impact at its output. The difference in the output value due to perturbation of input signal quantities is referred to as un-smooth error. Due to the nonlinearity of un-smooth operations, the dynamics of un-smooth errors cannot be captured using Taylor series expansion and first-order linear approximation as described in [12].

[1]Operation with n inputs and one output.

To illustrate the dynamics of propagating a small perturbation through un-smooth operations, two examples are considered as shown in Fig. 1. In order to keep the example simple, a two-input *min()* and a binary phase shift keying (BPSK) discriminator (discriminates between positive and negative input values) are considered. Consider two scenarios marked Case A and Case B according to the values taken by input signals. In Case A, the nominal values of the input signals for both examples are such that a small perturbation in its input does not produce a different output at the output of the un-smooth operator. In Case B for the BPSK operation case, if the magnitude of input perturbation results in a positive value and is as much as q_r in magnitude, the value of the signal at the output of the BPSK operation is V_2 instead of V_1. Similarly, in Case B for the *min()* operation example, the output signal y is assigned to x_2 nominally. In the presence of small perturbations, if the values taken by x_1 and x_2 are sufficiently far apart, y will be assigned the value taken by signal x_2, and therefore, there will be no un-smooth error at y. However, when the nominal values are close enough such that it is not deterministically possible to say which of the two signals are going to take the lesser value in the presence of quantization errors, there are chances of assigning the value of x_1 to y instead of x_2. The region of overlap in the error probability density functions (PDFs) associated with both x_1 and x_2 is shown in the shaded part of the PDF in Fig. 1.

III. Hybrid Approach

In this paper, a given signal processing system is represented by a directed graph consisting of nodes corresponding to each operation in the system and edges pointing in the direction of signal flow in the system. From the definition of un-smooth operation, any operation in the given system is classified as either smooth or un-smooth, and correspondingly, there are smooth and un-smooth nodes in the system graph. The hybrid approach requires grouping smooth nodes together and such groupings are referred to as clusters. Under these definitions, this section describes the hybrid approach.

During fixed-point simulation, all operations are simulated using a fixed-point library. This translates to calling the function emulating fixed-point behavior for each and every fixed-point operation in the system. If the system under consideration consists of only smooth operations, it is possible to avoid performing fixed-point simulation by using the analytical techniques discussed in [6]. But, in the presence of even one un-smooth operation, the lack of applicability of analytical methods makes it inevitable to use fixed-point simulation for the entire system.

If the nominal values are known *a priori*, it is possible to mimic the perturbation due to fixed-point quantization using PQN-based models. In the proposed technique, the focus is on using such analytical models for estimating quantization noise in order to minimize the number of times an operation is required to be simulated using fixed-point libraries. The pseudocode of the hybrid simulation algorithm is presented in Algorithm 1. This approach essentially consists of five steps enumerated as follows:

1) clustering of smooth-operations;

Algorithm 1 Accelerated Evaluation

```
1: \* clustering smooth operations *\
2: Identify un-smooth Boundaries;
3: $\mathbf{H}(\mathbf{C}, \tilde{\mathbf{V}}, \tilde{\mathbf{E}})$ = GetClusterGraph($G(V, E)$);
   \* Deriving SNS models and calculating Lifetimes *\
4: for all Cluster $C_i \in H$ do
5:     $C_i$.DeriveSNSModel();
6:     $C_i$.EstimateLifetimeValue();
7:     \* Default mode: Analytical *\
8:     $C_i$.SimMode = false;
9:     \* Contains no un-smooth errors *\
10:    $C_i$.ResetLifetimeCounter();
11: end for
12: \* Get sub-graph of each un-smooth operator *\
13: for all un-smooth operators $\tilde{V}_i \in H$ do
14:     \* Algorithm 1a *\
15:     $S(C, E)$ = GetUnsmoothSubgraph($\tilde{V}_i$, $H$);
16:     $\tilde{V}_i.S = S(C, E)$;
17: end for
18: \* Hybrid Simulation: *\
19: for all $n \in N_t$: Input test-vector do
20:     while Entire $\mathbf{H}(\mathbf{C}, \tilde{\mathbf{V}}, \tilde{\mathbf{E}})$ is not simulated do
21:         T = GetReadyNodes($H(C, \tilde{V}, \tilde{E})$);
22:         for all $t \in T$ do
23:             \* Check if the node t is an un-smooth operator *\
24:             if ($t_i \in \tilde{V}$) then
25:                 \* Algorithm 1b *\
26:                 SimMode = SelectiveSimulationMode($t, n, H$);
27:                 if SimMode == true then
28:                     \* Set subgraph of node t to simulation mode *\
29:                     \* Algorithm 1c *\
30:                     SetSimMode($t$);
31:                 end if
32:             else
33:                 \* Check if the cluster is in simulation mode *\
34:                 if $t_i$.SimMode == false then
35:                     \* Algorithm 1d *\
36:                     SNSErrorProp($t$);
37:                 else
38:                     \* Algorithm 1e *\
39:                     SimErrorProp($t, H$);
40:                 end if
41:             end if
42:         end for
43:     end while
44:     Mark $\mathbf{H}(\mathbf{C}, \tilde{\mathbf{V}}, \tilde{\mathbf{E}})$ as un-simulated;
45: end for
```

2) deriving single-noise-source (SNS) models for every cluster;
3) determine lifetimes of clusters;
4) identifying subgraph topologies;
5) performing selective simulation.

In the first step, the original directed signal flow graph $G(V, E)$ is transformed into a directed cluster graph $H(C, \tilde{V}, \tilde{E})$, where $\tilde{V}$ is the set of un-smooth nodes from the original graph $G(V, E)$, C is the set of clusters consisting of smooth operations, and $\tilde{E}$ are the edges connecting the clusters and un-smooth operation nodes with one another. In subsequent steps, analytical performance estimation models and the error life-time of each cluster are derived. The simulation subgraph (a subset of the clustered graph H) required for performing hybrid simulation on the event of an un-smooth error is obtained in the fourth step. The last step corresponds to performing the actual hybrid simulation phase where parts of the fixed-point system are simulated selectively.

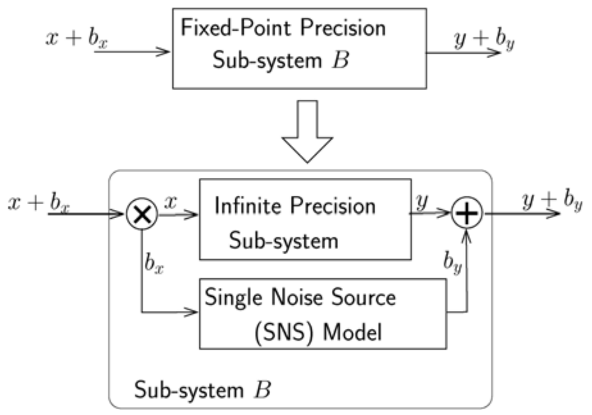
$x + b_x$
Fixed-Point Precision Sub-system B
$y + b_y$
$x + b_x$
x
Infinite Precision Sub-system
y
$y + b_y$
b_x
b_y
Single Noise Source (SNS) Model
Sub-system B

Fig. 2. Abstraction of the SNS model.

In the proposed hybrid simulation technique, the test vectors are processed one by one from a test suite consisting of N_t test vectors (or tokens). The simulation of a given signal flow graph is often considered a token processing system, where each input test case corresponds to one token. One token is consumed when one test case is simulated. Each input token activates a set of nodes T in the cluster graph H. These nodes consume the input token and produce one token each at their outputs. The new token generated by T nodes triggers another set of ready nodes. This process is repeated until 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 next input token is considered only after all nodes in the cluster graph are visited and evaluated either by simulation or by analytical techniques. The same procedure is repeated until all N_t tokens are processed.

In the rest of this section, the single noise source model, a PQN model based model for mimicking quantization errors at the output of a smooth cluster is first introduced. In Sections III-C and III-D, the evaluation of life-time and identification of simulation subgraph are discussed, respectively. In Section III-E, the algorithm that actually carries out selective simulation consistently with fixed-point simulation is described.

A. SNS Model

The SNS model is an abstraction to capture the stochastic behavior of errors generated by smooth quantizers. Perturbations generated by the SNS model are statistically equivalent to the fixed-point error obtained by simulation. A system implemented using fixed-point arithmetic (consisting of smooth operations only) and its SNS model abstraction is as shown in Fig. 2. In order to model the stochastic equivalence with fixed-point errors, the SNS model captures the spectral and noise distribution characteristics. The various components of the SNS model and their association to calculate the noise output are depicted in Fig. 3.

The analytical noise power evaluation [6] technique is applied to all clusters consisting of smooth operations. In this technique, an expression corresponding to the total output noise as a function of input noise variance and mean from each fixed-point arithmetic source is obtained analytically. The

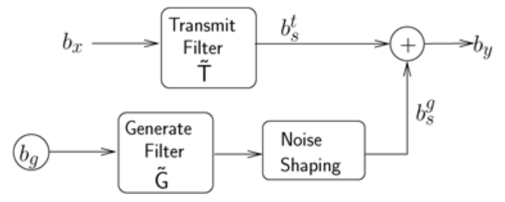

Fig. 3. Inside SNS model.

actual values of the noise are obtained by evaluating this expression (as opposed to fixed-point simulation) for various fixed-point word-lengths. Usually, the statistical quantity of interest is the total noise power due to quantization errors at the system output. An implicit assumption that the noise is normally distributed and is temporally uncorrelated is generally made. In [14, ch. 3], cases where such assumptions are not necessarily correct are presented. This is, indeed, the motivation for deriving the SNS model. In summary, the spectral parameters are required to express the quantization noise properties when the system under consideration is frequency selective (i.e., it has memory) and the distribution parameters need to be considered only if the quantization noise is being presented at the input of a level sensitive operator. It is worth noting that the un-smooth operation, as described in Section II, is a level sensitive operation.

Following the technique described in [15], it is possible to analytically derive the spectral parameters of the SNS model. In [14], [16], and [17], techniques for expressing the PDF by primarily qualifying it with the fourth moment of total quantization error are described. Similar to the technique described in [6], these techniques essentially require computation of the path function (transfer function in the case of LTI systems) of the path from all operations to the system output. The actual values, however, depend on the fixed-point formats assigned to each of the arithmetic operations. In this paper, the time spent on evaluating path functions is referred to as the preprocessing time (t_{pp}), and the time spent on evaluation of actual values is referred to as time spent on analytical evaluation (t_{ana}). Therefore, the calculation of actual parameters has to follow the application of preprocessing techniques. This is discussed at length in [14] and is omitted here for the lack of space. The technique for identification of all paths involving the preprocessing step is discussed in [7]. The generalization of the same is described in [6]. The algorithm adopted to find all paths in the system graph is described in [18].

The idea behind using this in the proposed hybrid approach is to reduce the computationally intensive task of fixed-point simulation to mere random number generation. In every iteration of the fixed-point simulation, the data from corresponding floating-point simulation is used as a reference value and the random number generated by the SNS model creates the perturbation effect as caused by errors due to fixed-point computation. Thus, with the help of the SNS model, quantization errors due to fixed-point operations are statistically mimicked without having to perform fixed-point simulation. When there are no un-smooth operations, the error generated at the output of a subsystem is propagated to the system output through other subsystems seamlessly.

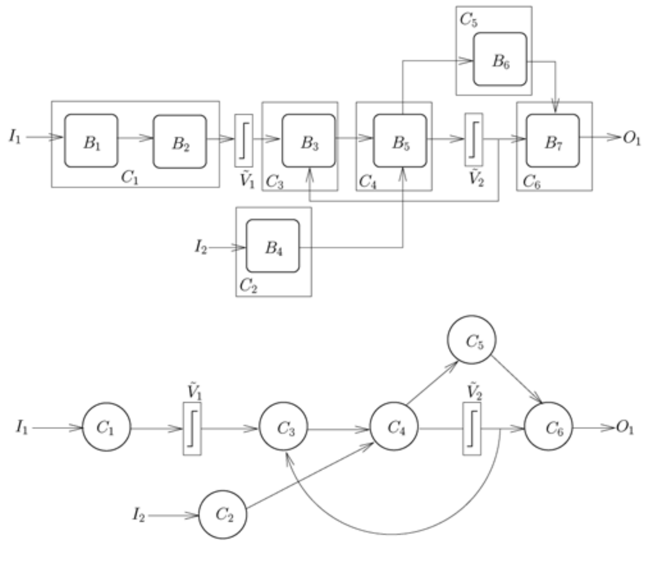

Fig. 4. Representative signal processing system.

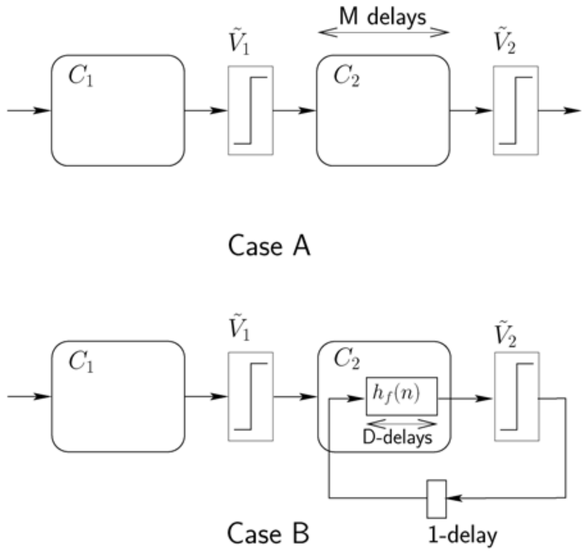

Fig. 5. Two topologies: Case A: feed-forward and Case B: feedback.

B. Clustering Smooth Operations

Consider an example of a signal processing system with two un-smooth operators and seven smooth blocks, as shown in Fig. 4. The first among preprocessing steps is to partition the given graph into smooth clusters separated by un-smooth operators. In the example considered, there are no un-smooth operators between blocks B_1 and B_2. Therefore, they can be clustered together into C_1. An erroneous un-smooth input to a smooth block 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 $\tilde{V}_2$ and $\tilde{V}_1$ although B_5 does not have any un-smooth input. Similarly, B_4, B_5, and B_6 cannot be combined together as it might be required to perform simulation of B_5 and B_6 due to an un-smooth error, whereas B_4 does not require simulation. The clustered graph obtained by combining smooth blocks is shown at the bottom in Fig. 4.

In this example, B_5 and B_6 are kept separate to form clusters C_5 and C_6. The cluster C_5 has two outputs, one of them feeds into the un-smooth operation $\tilde{V}_2$ and the other output feeds into cluster C_6. Combining B_5 and B_6, the new cluster (but now bigger) will be topologically no different from the cluster C_5 in the graph shown in Fig. 4. In order to use the proposed hybrid simulation algorithm, it is sufficient if all the un-smooth operations are separated from the smooth ones. Although keeping blocks B_5 and B_6 separate in the cluster graph rather than combining them can help a divide-and-conquer of the SNS evaluation step, this angle for optimization is not explored in this paper.

C. Lifetime of Un-Smooth Error

When an error is injected into a data path containing memory, it can influence the system output for more than one iteration. The number of successive iterations an un-smooth error affects the output of a smooth cluster beginning from the iteration in which it first occurred is referred to as its Lifetime. In other words, when an un-smooth error is injected into a smooth cluster, it requires simulation for Lifetime number of successive iterations. 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 the number of memory elements in the path between them. If a cluster has more than one input and output, each combination of input–output can have different values of Lifetime. In the case of a cluster with no memory, the un-smooth error does not affect successive iterations and its Lifetime is zero.

Considering the topology of the interconnect between clusters and decision operators, two unique scenarios are considered: one without any feedback containing the un-smooth operation in its path and the other case consisting of a feedback path with un-smooth node in it. These two scenarios are represented by two topologies in Fig. 5. In both cases, there are two smooth clusters (C_1 and C_2) and two un-smooth operators $\tilde{V}_1$ and $\tilde{V}_2$. The topology of the individual clusters (C_1 and C_2) could be either feed-forward (FF) or feedback (FB) in nature. In either case, the first cluster C_1 does not require simulation. In Case A, the second cluster needs simulation if an error occurs at the output of un-smooth operator $\tilde{V}_1$.

The Lifetime of the error injected into cluster C_2 depends on the path between its input (from $\tilde{V}_1$) and output (to $\tilde{V}_2$). If C_2 has a feed-forward topology, the effect of decision error at the input is flushed out after a deterministic number of samples. The Lifetime can be determined by just counting the number of delays (M) on the path from the input to the output of the second cluster C_2. On the other hand, if the cluster C_2 has a feedback structure such as an infinite impulse response filter, the error remains in the subsystem indefinitely with a diminishing effect. Then, 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 is negligible [19].

As discussed earlier, the technique used for calculating the pseudo-impulse response of recursive systems is described

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

Procedure 1a GetUnsmoothSubgraph(t, G)

```
1: \* Prune away all un-smooth operations except t *\
2: $G'$ = PruneNode($t$, $G$);
3: \* Identify all connecgted components in $G'$ *\
4: $H(C, E)$ = GetConnectedComponents($G'$, $t$);
5: Return $H(C, E)$;
```

in [6]. The algorithm proposed in [18] is used to identify all paths and cycles in the cluster graph. The complexity of identifying the Lifetime of an un-smooth error at the input of a smooth cluster is the same as identifying the longest delay path from the cluster input to its output.

The evaluation of Lifetime of clusters for Case B needs to consider two sources of un-smooth errors ($\tilde{V}_1$ and $\tilde{V}_2$), and the second source is a part of a feedback loop. An error at the output of the operator $\tilde{V}_2$ affects the output of the cluster C_2 in the next iteration. From Fig. 5, the error is held in the delay outside the cluster during the iteration in which the un-smooth error actually occurred. It has to be noted here that in any signal flow graph, a feedback loop must contain at least one delay element to avoid race conditions (sometimes also referred to as negative edge cycles [20]). So assume that one delay element outside the cluster causes no loss of generality. The value of the Lifetime for the feedback path of cluster C_2 when there is a decision output in the feed back path is equal to the path delay in the cluster plus the number of delay elements in the feed-back path. If the function $h_f(n)$ between the feed-back input to the output of cluster C_2 consists of a maximum of D delay elements and if there is one delay element on the feed-back path, the Lifetime of the cluster C_2 corresponding to the loop is $D + 1$.

D. Simulation Subgraph

Having determined the Lifetime of un-smooth errors across each input–output pair of all clusters, the next task is to identify the subgraph of every un-smooth operation in the cluster graph. This subgraph would be simulated in the event of an un-smooth error, occurring at the output of the corresponding un-smooth operation. The extent to which an un-smooth error affects the rest of the system depends on its locality in the system graph. 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 analyzing the cluster graph. The selective simulation is applied to every subgraph at the output of all un-smooth operations in the system. It is enough if the error propagation due to an un-smooth error is traced until the next un-smooth operation along the datapath in the direction of system output or the system output itself is reached.

The procedure for obtaining the subgraph of the node starting from the un-smooth operator node where the decision error has occurred is outlined in Algorithm 1a. The given system graph G is pruned off all un-smooth operations, except the given un-smooth operation t. From the remaining nodes in the graph, the subgraph is obtained as the connected component of the remaining nodes with the given un-smooth operation node t.

For the signal flow graph example considered in Fig. 4, subgraphs of nodes $\tilde{V}_1$ and $\tilde{V}_2$ are shown in Fig. 6. In

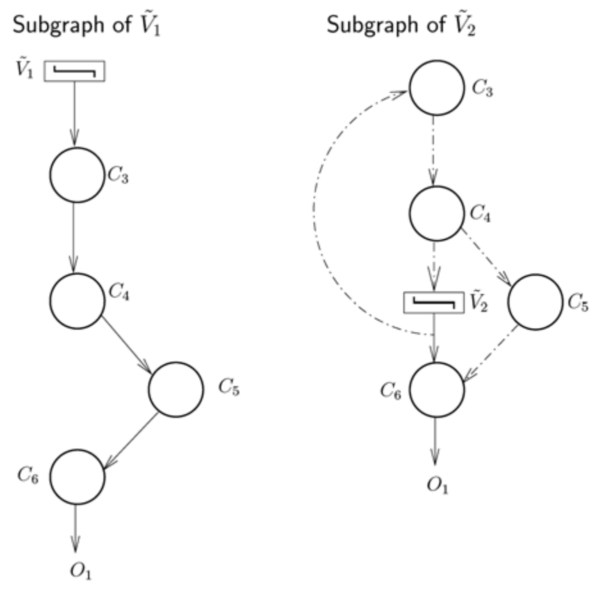

Fig. 6. Subgraph of nodes $\tilde{V}_1$ and $\tilde{V}_2$.

the case of $\tilde{V}_1$, the only other un-smooth operation $\tilde{V}_2$ is removed from the cluster graph. The clusters along path $C_3 \rightarrow C_4 \rightarrow C_5 \rightarrow C_6$ are a part of connected component in the cluster graph. In the case of an error at the output of un-smooth operation $\tilde{V}_1$, these clusters are simulated beginning with the iteration when the error occurred. In the case of $\tilde{V}_2$, there are two edges: one forward connecting to C_6 and the other backward edge connecting to C_3. Clusters C_4 and C_5 are connected to un-smooth operation $\tilde{V}_2$ through the back edge. When an un-smooth error occurs at the output of $\tilde{V}_2$, the nodes C_3, C_4, and C_5 would already be simulated by the time node $\tilde{V}_2$ is considered for simulation. 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 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 feedback loop, and hence does not affect the computations in the present iteration.

E. Hybrid Evaluation 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 these errors are such that they generate no un-smooth errors. Therefore, the counter to keep track of the Lifetime of the error in every cluster is initially reset to zero and the associated flag indicating 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.

1) *Checking for Un-Smooth Errors:* Algorithm 1b describes a way for checking whether an error can occur at the output of the un-smooth operator. Called from the hybrid simulation algorithm provided in Algorithm 1, this algorithm is executed in every simulation iteration of the hybrid simulation procedure. Here, the first step is to 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. Let $\mathbf{C}_t$ be the set of smooth clusters whose outputs are fed into the un-smooth operator t. More than one cluster feed into un-smooth operators such as *min()* or *max()*. This is obtained by calling the function *GetFeedinClusters*() with argument t on the graph H. For each cluster feeding into the un-smooth operation, the corresponding reference is obtained from the database if it is in the analytical mode in the current iteration. Otherwise, the value at the output of smooth clusters is obtained from the evaluation in the current iteration. In the analytical mode, during the iteration corresponding to the nth token, the nominal reference value vector $\mathbf{x}_t(n)$ corresponding to the output of the clusters $\mathbf{C}_t$ is obtained from the database. It would be perturbed by the quantization noise vector $\mathbf{b}_{x_t}$ due to smooth fixed-point quantization from respective clusters. Later, it is checked if this perturbation is large enough to cause an error by comparing the input values and the operation functionality by calling function *SusceptibletoError*(). If the conditions are indeed susceptible to cause an error, the actual value at the output of the un-smooth operation is evaluated using its function $t()$. If the output of the un-smooth operation matches the corresponding reference output, the SimMode continues to be false or is set to either true otherwise.

If some of the clusters are in the simulation mode, it is because of an un-smooth error elsewhere in the graph or in the previous iterations. Therefore, it is possible that the output statistics of the cluster is not a simple perturbation of the reference signal. Hence, even if one of the inputs is in simulation mode, the un-smooth operation has to be simulated. The SimMode flag is set to *true*, which indicates that the output of the un-smooth operation under consideration could be generating erroneous outputs. In this case, the un-smooth operation is actually simulated by calling the corresponding un-smooth function: $t()$ with values obtained by perturbation or obtained by fixed-point simulation $\hat{\mathbf{x}}_t$ is evaluated and the SimMode flag continues to be *true*.

Therefore, in scenarios such as Case A in Fig. 1, 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.

2) *Propagating Simulation Mode:* The correctness of the proposed hybrid algorithm depends on performing simulation of the subsystems whenever un-smooth errors are present. Only those subsystems that get affected by the un-smooth error shall be simulated. Such subsystems are already identified during the preprocessing phase. The occurrence of an un-smooth error means that an input that 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 the entire subgraph are potentially presented with a different input.

Such subsystems lie on the path starting from the node at which the un-smooth error occurring 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

Algorithm 1b SelectiveSimulationMode(t, n, H)

```
1:  \* Identify cluster at the input of un-smooth operator $t$ *\
2:  $\mathbf{C}_t$ = GetFeedinClusters($t$, $H$);
3:  SimMode = false;
4:  for all $C_i \in \mathbf{C}_t$ do
5:      if $C_i$.SimMode == false then
6:          \* Obtain $n^{th}$ reference sample *\
7:          $x_i(n)$ = GetReferenceSample($n$);
8:      else
9:          \* Obtain the $n^{th}$ sample from simulation *\
10:         $x_i(n)$ = GetValue($n$);
11:         SimMode = true;
12:     end if
13: end for
14: if (SimMode == false) then
15:     if t.SusceptibletoError($\mathbf{x}_t$) then
16:         $\mathbf{b}_{x_t}$ = GenerateRandomQNoise($\mathbf{C}_t$);
17:         $\hat{\mathbf{x}}_t = \mathbf{x}_t + \mathbf{b}_{x_t}$;
18:         if ($t(\hat{\mathbf{x}}_t) \neq t(\mathbf{x}_t)$) then
19:             SimMode = true;
20:         end if
21:     end if
22: else
23:     Evaluate $t(\hat{\mathbf{x}}_t)$;
24: end if
25: Return SimMode;
```

Algorithm 1c SetSimMode(t)

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

Algorithm 1d SNSErrorProp(C, n)

```
1: $b_i$ = GetInputQnoiseParams($C$);
2: \* Analytical Evaluation *\
3: $b_o$ = EvalOutputQNoise($C$, $b_i$);
```

to the subgraph of that un-smooth operator is marked *true*, as shown in Algorithm 1c. This means that the noise has to be propagated through all those nodes by simulation. Along with setting the flag, the Lifetime counter is set to the precomputed value as obtained in the preprocessing phase. This keeps a count of the number of successive tokens affecting the computation of the cluster. This value is already computed during the preprocessing phase.

3) *Evaluating Quantization Noise:* The idea of conditional simulation of subsystems 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 subsystems only when there is a deviation from the reference values caused by the occurrence of un-smooth errors.

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 subsystem

Algorithm 1e SimErrorProp(C, H)

```
1: $b_i$ = GetInputQnoiseParams($C$);
2: $b_o$ = Simulate($C, b_i$);  \* Evaluation by simulation *\
3: $C$.DecrementLifetimeCounter();
4: if $C$.GetCounterValue() == 0 then
5:     $C$.SimMode = false;
6: end if
```

and then evaluates the quantization noise at the output by using the function *EvalOutputQNoise*() as shown in Algorithm 1d.

When an un-smooth error occurs at the input of any cluster, it would be marked with a *SimMode* flag in Algorithm 1 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 Algorithm 1e, 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.

F. Hybrid Simulation Technique Effort

In the proposed hybrid approach, a finite amount of time is spent for the preprocessing overhead which includes identification of un-smooth operators, graph transformations, derivation of the SNS model for smooth clusters, and figuring out their *Lifetimes* and identifying their simulation subgraph. This is a one time effort and let t_{pp} be the time required to perform these preprocessing activities. Let t_{sim} be the time required for performing one iteration of the fixed-point simulation of a given signal processing system. If N_t is the total number of samples in a fixed-point simulation set, the total time for simulation T_{sim} is given as $T_{sim} = N_t \cdot t_{sim}$. Once the hybrid evaluation begins, parts of the system are simulated depending on the occurrence of un-smooth errors. The total time for executing all the test cases can, therefore, be written as

$$T_{hyb} = t_{pp} + (N_a \cdot t_{ana} + N_s \cdot \tilde{t}_{sim}) \tag{2}$$

where N_a is the number of simulation iterations in which there are no un-smooth errors. The error behavior in these iterations is consistent with the SNS model. The rest of samples $N_s = N_t - N_a$ are those in which at least one of the un-smooth operations causes an error. Thus, parts of the system need simulation depending on the point of occurrence of un-smooth error. While t_{ana} is the time required to generate samples of the actual random process, 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. As mentioned in Section III-A, t_{pp} is the time required to capture the path function of all paths in the smooth cluster required to define its SNS model.

The value of t_{ana} is very small in comparison as it involves generation of random numbers. The value of $\tilde{t}_{sim}$ has the worst case value equal to t_{sim}, which is the time required for one simulation iteration of the given system. Its average value influenced by the number of decision errors occurring and is also a characteristic of the input test vectors.

Typically, the size (N_t) of the fixed-point simulation set in a practical case is enormous. It is usually done so in order to cover all possible scenarios in which the signal processing system needs to be functionally tested. Performance evaluation of fixed-point systems for its accuracy requires to be performed repeatedly in the word-length optimization scenario. Several such simulations have to be performed to satisfactorily explore the word-length search space. To provide a qualitative estimate of the vastness of this search space, it has to be noted that the search space increases exponentially with every additional fixed-point operation in the system. The SNS model is obtained after the preprocessing step, it is reused in every iteration and for all different fixed-point word-length assignments.

Consider a typical word-length optimization scenario requiring N_{iters}. The value of N_{iters} can typically run to several tens and sometimes even hundreds even in relatively small cases. The time taken for fixed-point simulation and hybrid simulation during the entire process of word-length optimization is given, respectively, as

$$\begin{aligned} T_{\text{sim}}^{\text{iters}} &= N_{\text{iters}} \cdot (N_a + N_s) \cdot t_{\text{sim}} \\ T_{\text{hyb}}^{\text{iters}} &= t_{pp} + N_{\text{iters}} \cdot (N_a \cdot t_{\text{ana}} + N_s \cdot N_{\text{sim}}). \end{aligned} \tag{3}$$

If the time spent on preprocessing is amortized for N_{iters}, the time t_{pp} spent on preprocessing is negligibly small. On a per-iteration basis, the total time spent on simulation by using the proposed hybrid approach can be approximated as

$$T_{hyb} \approx N_a \cdot t_{ana} + N_s \cdot \tilde{t}_{sim}. \tag{4}$$

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

$$\begin{aligned} IF &= \frac{T_{sim}}{T_{hyb}} \\ &= \frac{(N_a + N_s) \cdot t_{sim}}{N_a \cdot t_{ana} + N_s \cdot \tilde{t}_{sim}}. \end{aligned} \tag{5}$$

In order to keep the system functional, the number of decision errors due to quantization noise is not very common and therefore it is typical that $N_s << N_a$. Therefore, the typical values of IF are large. The maximum IF value is limited by the generation of random numbers and checking for un-smooth errors. It is given by

$$IF_{max} = \frac{t_{sim}}{t_{ana}}. \tag{6}$$

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. Also, as the precision of the fixed-point numbers increases, the number of un-smooth errors tends to decrease, thereby improving the IF.

G. Equivalence With Fixed-Point Simulation

In the Hybrid approach, the noise at the output of a subsystem 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. The errors at the output of the un-smooth operator can be estimated with the knowledge of the signal and noise PDF as discussed in [11]. 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 in the place of 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 paper, 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.

In order that the hybrid technique converges with the fixed-point simulation, it is essential that there are fixed-point errors spanning the entire range of quantization noise values at the source of every fixed-point error. This is possible by having a large number of samples and therefore, the convergence of error metric values obtained by hybrid simulation and fixed-point simulation largely depends on the number of points used for simulation.

IV. Experiments

To show the effectiveness of the hybrid approach, the results obtained by applying the proposed technique on some synthetic and practical examples are presented in this section. To begin with, a synthetic example is considered to illustrate the feed-forward topology shown in Fig. 5. To illustrate the case where decision output is fed-back, the classical decision feedback equalizer is considered. Then, the hybrid technique is applied to edge detection algorithm which uses morphological operators [21] to illustrate its applicability in the presence of *min()* and *max()* operations. The selective spanning with fast enumeration (SSFE) algorithm [22] is then considered to illustrate the effect of number of un-smooth operations on the effectiveness of the hybrid simulation technique.

Apart from the improvement factors obtained, the proposed hybrid simulation technique is essentially a faster alternative to 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 obtained. The results presented in each of the examples include plots and figures to show the statistical equivalence between the proposed hybrid technique with fixed-point simulation. The statistical equivalence between the two approaches is subject to small errors due to the limited number of points considered

TABLE I

Preprocessing Time and SNS Model Parameters

Expt.	$C_{\#}$	t_{pp}	Kurt	Spectrum
FIR-bank	C_1	0.55	2.9	17 tap, low pass, cut-off 0.5
	C_2	0.55	2.9	17 tap, low pass, cut-off 0.5
Edge Detect	C_1	0.44	2.7	2-D filter: $\begin{matrix} 0.25 & 0.5 & 0.25 \\ 0 & 0 & 0 \\ -0.25 & -0.5 & -0.25 \end{matrix}$
DFE	C_1	13.6	2.5	fwd taps: $0.7987 + 0.0021i$, $-0.2172 - 0.0688i$
	C_2		1.8	bwd tap: $-0.2925 + 0.0437i$
SSFE	C_1	0.31	2.7	*uncorrelated in time/ white*
	C_2	0.23	2.6	*un-correlated in time/ white*
	C_3	0.18	2.41	*un-correlated in time/ white*
	C_4	0.15	1.8	*un-correlated in time/ white*
	C_5	8.69	12.1	*un-correlated in time/ white*

for classical simulation. These errors tend to converge as the number of simulation points increases.

A. Experimental Setup

The experiments in this paper to show case the proposed hybrid simulation approach are set up in the MATLAB environment. A subset of the fixed-point library of MATLAB to simulate only the fractional bits is used for fixed-point simulation. ID.Fix [23], a tool being developed as a part of the GeCoS framework for automatic fixed-point conversion, is used to perform preprocessing including semantic analysis and expression for calculating the noise power. The work in this paper adds to the work already present as a part of the ID.Fix tool. The MATLAB scripts associated with each of the experiments were executed on a local PC/workstation.

A summary of the SNS parameters used is given in the Table I. The ID.Fix [23] tool accepts high level code in MATLAB or C describing signal processing functions and performs semantic analysis, builds signal flow graphs and other semantic analyses of the code. The time shown as t_{pp} corresponds to the analysis of the signal flow graph and generating source code to calculate the noise power at the output of the given function. It may be observed that for many examples in this paper, the time consumed for preprocessing is negligibly small, especially given that it is performed only once. Two exceptions may be noted for the DFE experiment and cluster C_5 of SSFE. It may be noted that these functions involve time varying systems, and hence, a large ensemble of values (in double floating point precision) has to be considered [6] to evaluate the noise power. In ID.Fix, MATLAB is used to perform the statistical analysis and hence the long time. Also, in the case of the DFE experiment, clusters C_1 and C_2 have been combined together to form one single smooth block; the imaginary and real parts of the noise parameters are evaluated separately. The high Kurtosis value of the cluster C_5 of SSFE can be attributed to the absolute and squaring function. This nonlinearity is also the reason for its long preprocessing times.

B. Feed-Forward Example

Consider the application of the hybrid technique on the feed-forward topology (Case A) shown in Fig. 5. The smooth clusters C_1 and C_2 are essentially linear filters. The quantization noise generation characteristics of each of these filters

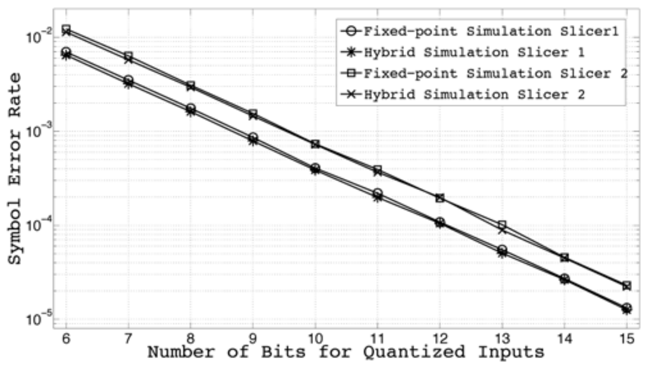

Fig. 7. Feed-forward case: un-smooth error rates.

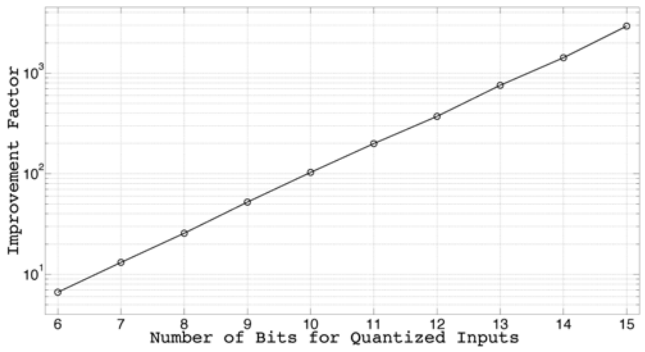

Fig. 8. Feed-forward case: evolution of IF.

are obtained by the application of the SNS model. A 17-tap low-pass hamming filter with a normalized cutoff frequency of 0.5 is chosen for both C_1 and C_2. Filter coefficients are quantized with 24 bits in its precision. The errors obtained at the output of both un-smooth operators by simulation and by the hybrid technique for various input bit-widths are shown in Fig. 7. The input word-length 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. In this experiment, 10^7 samples are used for performing both fixed-point and hybrid simulation. Relative error values obtained by hybrid simulation are as low as 2.2%.

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 Fig. 8. It can be seen that the improvement factor indicates several orders of magnitude speedup in the time required for carrying out simulation. The speed-up factor grows in magnitude as the quantization noise reduces. A summary of hybrid simulation parameters with 8 bits input quantization is in Table II.

C. Decision Feed-Back Equalization

Decision feedback equalizer (DFE) is a popular adaptive equalization technique used in various scenarios requiring error recovery. The block diagram of this DFE considered in this paper is shown in Fig. 9. It essentially consists of two arithmetic blocks, the feed-forward and the feedback blocks marked as clusters C_1 and C_2. Both these blocks are essentially tapped-delay lines whose weights are adapted according to the least-mean-square equalization algorithm by taking into consideration the values stored in the registers of the delay line in both feed-forward and feedback blocks. The decision error is fed back into the DFE through the feedback path. Both filters are essentially time varying in nature, and hence, the SNS parameters are measured after the training period is over. Table I describes the average values of the filter corresponding to the SNS model. Since both clusters are within themselves feed-forward, the spectral parameters of SNS correspondingly require only as many taps as the feed-forward filter structure. Both clusters are time varying and hence the preprocessing times for both clusters are nearly the same as most time is spent in calling and executing MATLAB routines and reading the file for accessing stored double precision values.

Fig. 10 shows the equivalence between fixed-point simulation and the hybrid approach which makes use of the single noise source model. The quantization noise generated 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 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 Fig. 11, the improvement factors obtained under various channel SNR conditions are 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 require simulation decreases, thereby contributing to an increase in the IF. This trend is also seen across varying channel noise. That is, the

TABLE II
SIMULATION RUN TIMES (ROUNDED TO NEXT INTEGER) AT 8-BITS PRECISION

Experiment	N_t	N_s	t_{fp}	$N_s \cdot \bar{t}_{sim}$	$(N_a) \cdot t_{ana}$
FIR-bank	10^7	3685	57	2.02	0.434
Edge Detect	512×512	678	454	1.22	0.22
@ 9 dB DFE	$3 \cdot 10^4$	501	11	0.14	$5 \cdot 10^{-3}$
@23 dB SSFE$^{1\ 1\ 1\ 4}$	$6.5 \cdot 10^4$	1233	159	1.1	0.7
@23 dB SSFE$^{1\ 1\ 2\ 4}$	$10.5 \cdot 10^4$	2624	279	1.9	1.3
@23 dB SSFE$^{1\ 2\ 2\ 4}$	$18.5 \cdot 10^4$	4867	454	3.44	2.8

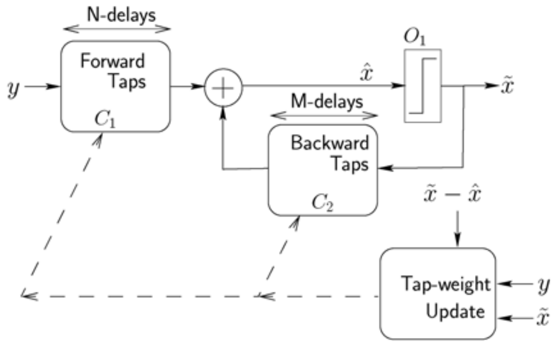

Fig. 9. Decision feedback equalizer block diagram.

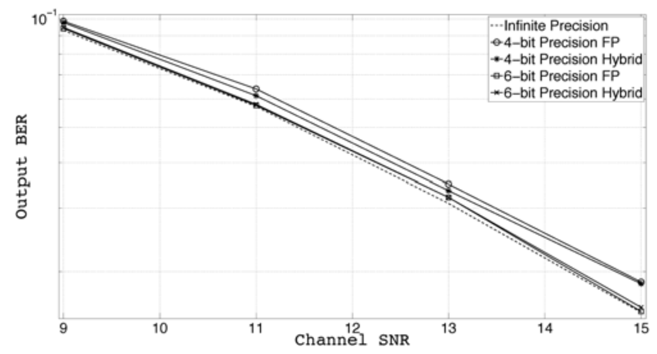

Fig. 10. DFE: hybrid simulation equivalence with fixed-point simulation.

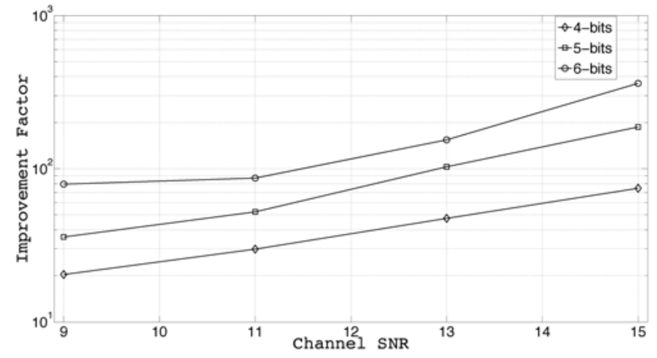

Fig. 11. DFE: evolution of IF.

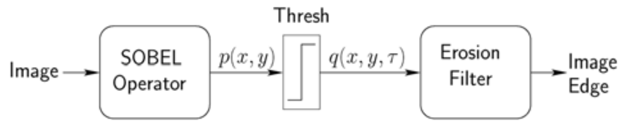

Fig. 12. Block diagram of SOBEL edge detector.

improvement factor increases with decreasing noise in the channel. The IF values achieved indicate speedup as high as two orders of magnitude.

In Table II, a summary of hybrid simulation parameters is provided for simulation under 9-dB channel noise conditions. The number of simulations carried out due to un-smooth errors persist for many more iterations. Under the conditions considered 501 samples were simulated and 199 iterations among them have occurred due to previous un-smooth operations lingering in the feed-back loop.

D. Edge Detection

Detection of edges in a given image is a problem very frequently encountered in many image processing applications. A popular application of edge detection is to sharpen or restore the quality [24] of blurred images. In this experiment, one of the popular schemes which uses the SOBEL edge detector [25] followed by thresholding and morphological Erosion is applied to the input image in order to identify the edges. The schematic of the edge detection algorithm is shown in Fig. 12.

The SOBEL edge detector is essentially a 2-D linear high-pass filter and consists of smooth arithmetic operations. The quantization noise due to fixed-point effects of the SOBEL operator 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.

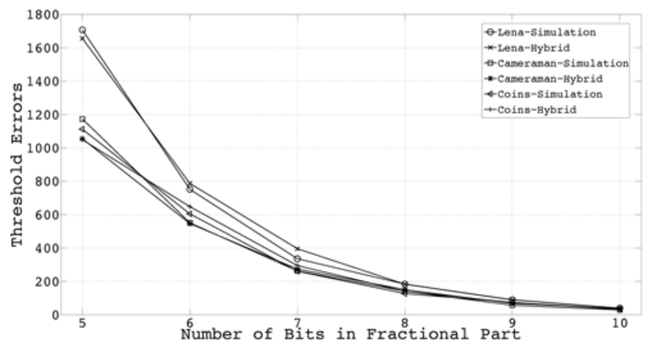

Fig. 13. Edge detection: errors comparison.

The term $p(x, y)$ corresponds to the intensity value of the pixel at the position (x, y)[3] and τ is the threshold level that 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 bidimensionality of this kernel makes the edge detection sensitive to edges aligned in both vertical and horizontal detections. The threshold and the morphological filter are both un-smooth. Therefore, a double-precision simulation of the test images is carried out, and the values at the output of the filter and the threshold operator are stored for reference.

Three representative image test cases: *Lena*, *Cameraman*, and *Coins* that are popular in the image processing domain are considered inputs for this experiment. Edge detection is performed in an image restoration scenario. Images of old manuscripts or portraits are smudged due various reasons including aging. This is often which is modelled as a blur due to ageing. Therefore, the sample images are first filtered through a Gaussian low-pass filter to emulate the blurring effect before passing it through the edge detection scheme proposed in Fig. 12. The plots in Fig. 13 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 close to one another with very little difference between results obtained by fixed-point simulation for various fixed-point precisions. The maximum relative error between the results obtained between fixed-point and Hybrid simulation is observed to be about 6%, thus validating the statistical equivalence of the hybrid approach with fixed-point simulation for the case of this experiment.

Fig. 14 shows the improvement factor obtained for various test cases and different levels of quantization of the unit normalized image input signal. As the number of precision bits increases, 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 indicates a maximum speedup of three orders of magnitude. A

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

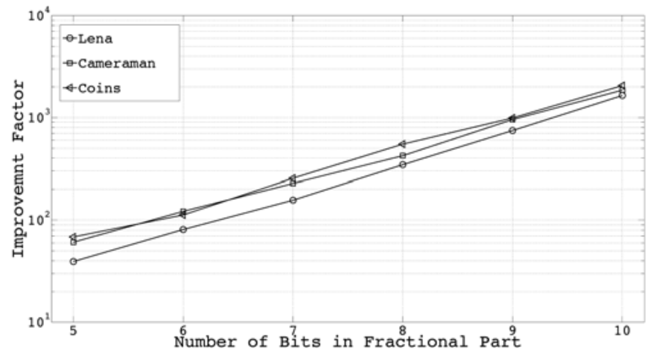

Fig. 14. Edge detection: IF.

summary of the hybrid simulation parameters obtained for a large 512×512 dimension *Lena* image is provided in Table II. The number of un-smooth errors are few and the time taken for analytical evaluation is much lower than the time required by simulation.

E. MIMO Decoding With SSFE

Multiple input multiple output antenna (MIMO) system for wireless communication promises improved communication efficiency [26] and has been widely studied and used in practical systems. The SSFE [22] 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 cancelation techniques.

The schematic in Fig. 15 shows the data-flow of the SSFE algorithm in the [4 2 1 1] configuration for a 4×4 MIMO receiver scheme. Here, each node corresponds to an un-smooth operation (a QAM slicer) and each of the edges corresponds to a smooth cluster. For the purpose of illustration, one branch of the SSFE diagram in Fig. 15 is shown in Fig. 16. The arithmetic operations involved in each cluster between slicers essentially correspond to inverting the R matrix obtained by QR-decomposition of the 4×4 channel matrix H. The smooth cluster C_5 corresponds to finding the sum of absolute Euclidean distance between the decoded value and the discriminated symbol. One value Z_i is computed per branch and finally the branch i such that $Z_i = min(\mathbf{Z})$ is chosen. Here again, the *min()* operation is un-smooth.

The [4 2 1 1] configuration indicates that four neighbouring constellations are explored and hence there are four branches in the corresponding SSFE tree diagram shown in Fig. 15. Similarly, two neighbouring symbols are explored at the output of slicers sl_3 and sl_2. This is indicated by two edges from node sl_3 in the SSFE tree diagram. In slicers sl_2 and sl_1, only the nearest symbol is considered, and therefore, it is indicated by just one edge in the SSFE tree diagram. 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.

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 is white and shaped by the number of errors added. A double-precision simulation is carried out on all input test cases, and the values at the input of every un-smooth operation are 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.

Fig. 17 shows the degradation of bit error rate (BER) with decreasing channel noise in the case of double precision simulation, fixed-point simulation, and the hybrid technique for the [4 2 1 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

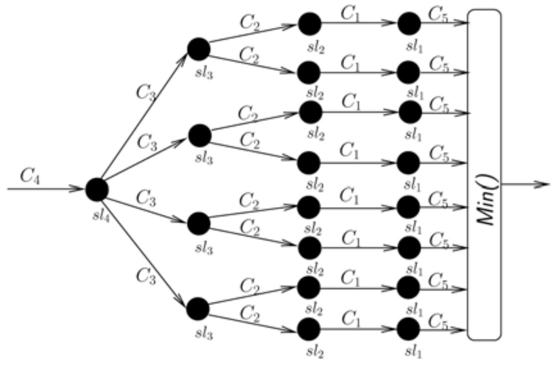

Fig. 15. SSFE data flow with configuration [4 2 1 1].

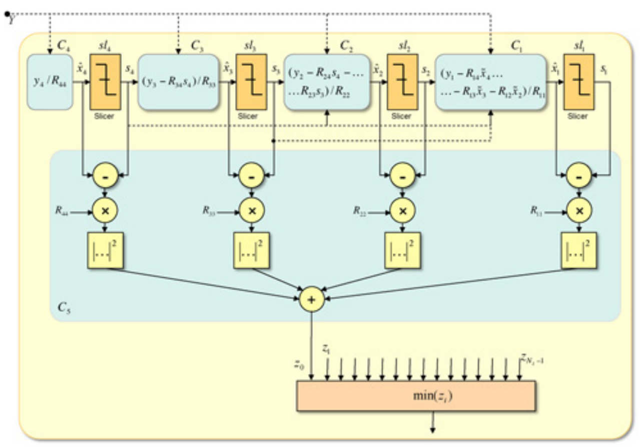
y
C_4
sl_4
C_3
sl_3
C_2
sl_2
C_1
sl_1
y_4/R_{44}
$(y_3 - R_{34}s_4)/R_{33}$
$(y_2 - R_{24}s_4 - \dots R_{23}s_3)/R_{22}$
$(y_1 - R_{14}\tilde{x}_4 \dots - R_{13}\tilde{x}_3 - R_{12}\tilde{x}_2)/R_{11}$
Slicer
R_{44}
R_{33}
R_{22}
R_{11}
C_5
z_0
z_1
z_{N_s-1}
$\min(z_i)$

Fig. 16. SSFE data flow model and associated smooth clusters.

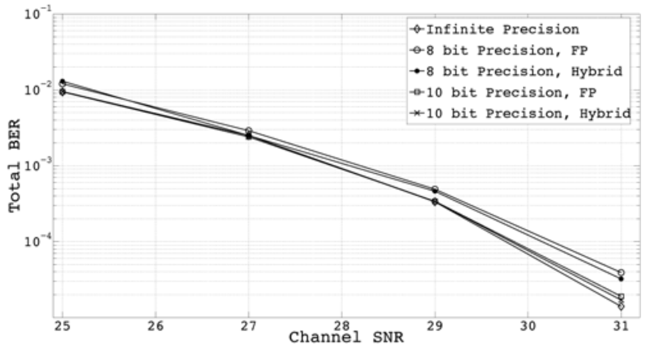

Fig. 17. BER degradation in SSFE for configuration [4 2 1 1].

TABLE III
COMPARATIVE STUDY OF VARIOUS SSFE CONFIGURATIONS AND THEIR RUN TIMES

SNR	$m = [1,2,2,4]$ (# 16)			$m = [1,1,2,4]$ (# 8)			$m = [1,1,1,4]$ (# 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**

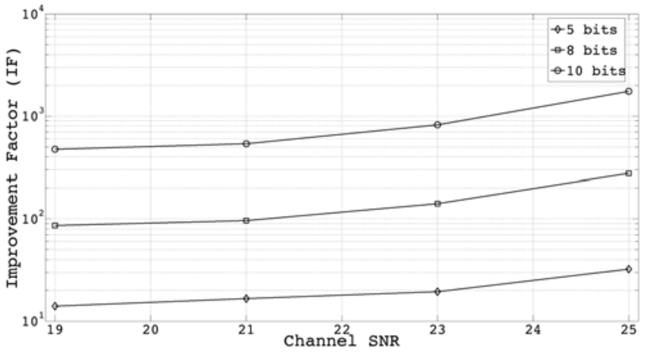

Fig. 18. Improvement factor for SSFE configuration [4 2 1 1].

of 8 bits and 10 bits are considered for comparison purposes. The data obtained indicate that the maximum error is 10% in the case of 8-bit 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 Fig. 18 for three different precision assignments: 5 bits, 8 bits, and 12 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 Fig. 18 is an indicator of the improvement that can be obtained in the case of low BER simulations.

A sample compilation of the simulation analysis parameters is shown for three different configurations of the SSFE in Table II. The SSFE configurations are shown as superscripts and the values in this table are obtained under 23-dB channel SNR conditions and all the fractional parts are truncated to 8 bits. The number of un-smooth errors is also correlated to the number of un-smooth operations. When there are many un-smooth operations, 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 operations with 14-bit precision fixed-point numbers. The results are summarized in Table III. 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 operations at the antenna 4 remains a constant, but are varied for other antennas. With increase in the number of un-smooth operations, the number of smooth clusters also increases. 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 result in increased time for performing Hybrid simulation. Thus, although the numerator of the improvement factor in (5) increases, there is also a corresponding increase in the denominator.

The results shown in Table III 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 (5). This trend can also be observed in Table II for different configurations. 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 operations.

V. CONCLUSION

The problem of evaluating the loss in performance of fixed-point signal processing systems that consist of un-smooth operations is the focus of this paper. A hybrid technique for accelerated simulation of the fixed-point system is proposed in the presence of un-smooth operations. This technique makes use of the classification of operators as smooth or un-smooth and uses the analytical SNS model obtained by using the analytical techniques discussed in [6] 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 selectively simulated only when un-smooth errors occur and not otherwise. Thus, the effort for fixed-point simulation is greatly reduced.

The preprocessing overhead consists of deriving the SNS model, and it is often small in comparison to the time required for fixed-point simulation. The advantage of using the proposed technique is that the user need not spend time on characterizing the nonlinearities associated with un-smooth operations.

The proposed hybrid algorithm is capable of working with topologies with feedback, unlike the completely analytical technique for un-smooth operations proposed in [11]. Several examples from general signal processing, communication, and image processing domains are considered for evaluation of the proposed hybrid technique. The acceleration obtained is quantified as an improvement factor. Very high improvement factors indicate that the hybrid simulation is several orders of magnitude faster than classical fixed-point simulation.

ACKNOWLEDGMENT

The authors would like to thank N. Simon, Engineer in the CAIRN Team, for helping them with measuring the preprocessing times.

REFERENCES

[1] G. A. Constantinides and G. Woeginger, "The complexity of multiple word-length assignment," *Appl. Math. Lett.*, vol. 15, no. 2, pp. 137–140, 2002.

[2] M.-A. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 2. 2002, pp. II-612–II-615.

[3] S. Kim and W. Sung, "Fixed-point-simulation utility for C and C++based digital signal processing programs," in *Proc. Rec. 28th Asilomar Conf. Signals Syst. Comput.*, vol. 1. 1994, pp. 162–166.

[4] Mathworks, "*Fixed-Point Blockset User's Guide (ver. 2.0)*," 2001.

[5] M. Willems, V. Bursgens, T. Grotker, and H. Meyr, "Fridge: An interactive code generation environment for HW/SW codesign," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Process.*, vol. 1. Apr. 1997, pp. 287–290.

[6] R. Rocher, D. Menard, O. Sentieys, and P. Scalart, "Analytical approach for numerical accuracy estimation of fixed-point systems based on smooth operations," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 10, pp. 2326–2339, Oct. 2012.

[7] D. Menard, R. Rocher, and O. Sentieys, "Analytical fixed-point accuracy evaluation in linear time-invariant systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 1, pp. 3197–3208, Nov. 2008.

[8] G. Caffarena, C. Carreras, J. López, and A. Fernández, "SQNR estimation of fixed-point DSP algorithms," *EURASIP J. Adv. Signal Process.*, vol. 2010, pp. 21:1–21:12, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1155/2010/171027.

[9] B. Widrow and I. Kollár, *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge, U.K.: Cambridge Univ. Press, 2008.

[10] K. Parashar, R. Rocher, D. Menard, and O. Sentieys, "Analytical approach for analyzing quantization noise effects on decision operators," in *Proc. IEEE Int. Conf. Acoustics Speech Signal Process.*, Mar. 2010, pp. 1554–1557.

[11] A. Chakari, K. Parashar, R. Rocher, and P. Scalart. (2012, Oct.). "Analytical approach to evaluate the effect of the spread of quantization noise through the cascade of decision operators for spherical decoding," in *Proc. DASIP* [Online]. Available: http://hal.inria.fr/inria-00534522/en/

[12] G. Constantinides, "Perturbation analysis for word-length optimization," in *Proc. 11th Annu. IEEE Symp. Field Program. Custom Comput. Mach.*, Apr. 2003, pp. 81–90.

[13] D. Menard and O. Sentieys, "Automatic evaluation of the accuracy of fixed-point algorithms," in *Proc. DATE*, Mar. 2002. pp. 529–535.

[14] K. Parashar, *System Level Approaches for Fixed-Point Refinement of Signal Processing Algorithms*. Ph.D thesis, Univ. de Rennes, Dec. 2012. [Online]. Available: http://hal.inria.fr/docs/00/78/38/06/PDF/Thesis_KarthickParashar.pdf

[15] K. Parashar, D. Menard, R. Rocher, and O. Sentieys. (2010, Aug.). "Estimating frequency characteristics of quantization noise for performance evaluation of fixed point systems," in *Proc. EUSIPCO*, pp. 552–556 [Online]. Available: http://hal.inria.fr/inria-00534524/en/

[16] K. Parashar, D. Menard, R. Rocher, and O. Sentieys, "Shaping probability density function of quantization noise in fixed point systems," in *Proc. ASILOMAR*, 2010, pp. 1675–1679.

[17] R. Rocher and P. Scalart, "Noise probability density function in fixed-point systems based on smooth operators," in *Proc. DASIP*, 2012, pp. 1–8.

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

[19] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[20] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, 1st ed. New York, NY, USA: Wiley-Interscience, 1999.

[21] J. Lee, R. Haralick, and L. Shapiro, "Morphologic edge detection," *IEEE J. Robot. Autom.*, vol. 3, no. 2, pp. 142–156, Apr. 1987.

[22] M. Li, B. Bougard, E. Lopez, A. Bourdoux, D. Novo, L. Van Der Perre, *et al.*, "Selective spanning with fast enumeration: A near maximum-likelihood mimo detector designed for parallel programmable baseband architectures," in *Proc. IEEE Int. Conf. Commun.*, May 2008, pp. 737–741.

[23] D. Menard, "Id.fix: Infrastructure for the design of fixed-point systems," in *Proc. Univ. Booth IEEE/ACM Conf. Design, Autom. Test Eur.*, 2011. [Online]. Available: http://idfix.gforge.inria.fr.

[24] F. Y. Shih, *Image Processing and Mathematical Morphology: Fundamentals and Applications*, 1st ed. Boca Raton, FL, USA: CRC Press, 2009.

[25] T. Peli and D. Malah, "A study of edge detection algorithms," *Comput. Graph. Image Process.*, vol. 20, no. 1, pp. 1–21, 1982.

[26] K. Peppas, F. Lazarakis, D. Axiotis, T. Al-Gizawi, and A. Alexandridis, "System level performance evaluation of MIMO and SISO of DM-based wlans," *Wireless Netw.*, vol. 15, pp. 859–873, Jul. 2009.

Karthick Nagaraj Parashar received the M.S. degree from the Indian Institute of Technology Madras, Chennai, India, in 2008, and the Ph.D. degree from INRIA-Rennes Bretagne Atlantique, Rennes, France, in 2012.

He is currently a Post-Doctoral Researcher with the Circuits and Systems Group, Imperial College, London, U.K. He was with the CAIRN Team while pursuing the Ph.D. degree. He has also been a Research and Development Engineer with Synopsys, Bangalore in the past. His current research interests include numerical error analysis, fixed-point conversion, design of FPGA-based accelerators, and mapping signal-processing algorithms to hardware and embedded software architectures.

Daniel Menard (M'01) received the Ph.D. and HDR (habilitation to conduct research) degrees in signal processing and telecommunications from the University of Rennes (ENSSAT), Rennes, France, in 2002 and 2011, respectively.

From 2003 to 2012, he was an Associate Professor with the Department of Electrical and Computer Engineering (ECE), Engineering School, ENSSAT. He was also a member of the IRISA/INRIA Laboratory. He is currently a Professor with the Department of ECE, Engineering School, Institut National des Sciences Appliquées de Rennes, Rennes. He is also a member of the IETR/CNRS Laboratory. His current research interests include implementation of signal processing and mobile communication applications in embedded systems, fixed-point conversion, embedded software, low-power architectures and arithmetic operator design, video compression, and visual sensor networks.

Olivier Sentieys (M'94) joined the University of Rennes (ENSSAT) and IRISA Laboratory, Lannion, France, as a Full Professor of Electronics Engineering in 2002. He leads the CAIRN Research Team common to INRIA (National Research Institute in Computer Science) and IRISA Laboratory, where he is the Head of the Computer Architecture Department. Since September 2012, he has been on secondment at Equipe CAIRN, INRIA, Lannion, France, as a Senior Research Director. His research activities are in the two complementary fields of embedded systems and signal processing. He has authored or co-authored more than 150 journal publications or peer-reviewed conference papers. He holds five patents. His current research interests include numerical accuracy analysis, computer-aided design tools for floating-point to fixed-point conversion, low-power and reconfigurable systems-on-a-chip, design of wireless communication systems, and cooperation in mobile systems.