Brief Contributions

Accuracy vs. Precision in Digital VLSI Architectures for Signal Processing

Cesare Alippi, *Member*, *IEEE*,
and Luciano Briozzo, *Member*, *IEEE*

Abstract—The paper provides a sensitivity analysis to measure the loss in accuracy induced by perturbations affecting acyclic computational flows composed of linear convolutions and nonlinear functions. We do not assume a large number of coefficients or input independence for the convolution module, nor strict requirements on the nonlinear function. The analysis is tailored to digital VLSI implementations where perturbations, associated with data quantization, affect the device inputs, coefficients, internal values, and outputs. The sensitivity analysis can be used to measure the loss in accuracy along the computational chain, to characterize the tolerated perturbations, and to dimension the whole architecture.

Index Terms—Finite precision representation, neural networks, NSR, sensitivity analysis.

1 Introduction

THE problem of evaluating the effects caused by finite precision on a computational flow plays a relevant role in the design and configuration of an algorithm specific architecture before its final implementation. In general, a compromise between the hardware compactness, in terms of silicon area, and the loss in accuracy at the device output caused by finite representation is required. Normally, this aspect is tackled by first configuring the architecture and the word precision and, then, testing the loss in accuracy at the device output with respect to the ideal algorithm. The architecture and precision are then repeatedly modified until a suitable compromise between hardware cost and accuracy is achieved. In some other cases where we are interested in representing some critical events, the signal itself is used to determine the precision resolution [1].

Here, we propose a sensitivity analysis which correlates, at a behavioral level, the influence of finite precision on the device's accuracy. The sensitivity analysis is twofold: It provides a measure of robustness for the algorithm once affected by noise or structural modifications (e.g., caused by a gradual aging effect); it computes the degradation in accuracy consequent to a finite precision implementation; and, then, it uses the results backward to guide the dimensioning of the whole architecture. The attention of this paper will be focused on the last issue by specializing general results to a particular digital VLSI implementation. In such a case, perturbations or *noise* mainly refer to two cases: external noise affecting input data and quantization noise.

In this paper, we consider computational blocks composed of operators such as addition, multiplication, linear convolution, and nonlinear transformation. These elements constitute building blocks for most of classic processing techniques (e.g., the ones dealing with convolutions, FFT, Hartley transforms [2]) and emerging ones as neural networks [3] and wavelet transforms [4].

The feedforward neural network [3] structure, which consists of cascades of nonlinear convolvers and which may receive dependent inputs, is a suitable example of sophisticated computational flows. When the activation function is linear, a neuron degenerates to a linear convolver and the network to a cascade of linear convolvers. We consider feedforward neural networks as a case study; the results will be, in any case, general and it is reasonably simple to specialize them to a specific application.

The problem of evaluating the effects caused by finite precision in networks receiving binary inputs and characterized by hard limited functions is not new. Pioneering research in this direction has been conducted in [5] by assuming small perturbations affecting the network's coefficients (or weights); this constraint has been relaxed in [6]. Results have been further extended in [7] to deal with real inputs and continuous activation functions. Other related interesting analyses can be found in [8], [9], where the attention is focused on quantization effects.

All the above mentioned authors consider a large fan-in for the non linear neural convolver and invoke the central limit theorem. As a consequence, they can assume Gaussian distributions for errors, convolution values, and outputs. Such a hypothesis is a panacea since we do not have to worry about dependency on inputs and their statistical distribution. In contrast to the above authors' approach, we remove the very restricting hypothesis of considering only very large networks, since many real image/signal processing applications deal with a reduced number of convolution coefficients. Moreover, the inputs are generally correlated, e.g., consider a convolver which processes inputs coming from common data. Even when an application requires a large network, it may be necessary to prune unnecessary weights to improve performance [10]; this generates sparse topologies and the Gaussian hypothesis may be not satisfied locally, hence reducing the effectiveness of the above-mentioned models.

Our framework supports the network ensemble case which is particularly relevant if the goal is either to develop a general purpose architecture or to permit some on-line weights adjustment. On the other hand, it can also be used to develop dedicated devices which are primarily designed for specific application for which the optimal weights are given and computed off-line. For such an application we assume a tailored analysis to exploit a priori knowledge and improve the model effectiveness.

In the following analyses, for each computational module, we will consider two distinct computational flows: the ideal one, in which the computation is error free, and the real one, implemented by an error affected device, e.g., a physical device implementing the computation with finite precision.

As a natural measure for the loss in accuracy caused by a generic perturbation in a specific point of the computational chain, we consider the Noise to Signal Ratio *NSR* defined as the ratio of the variance of the perturbation—or noise—σ_n^2 to that of the error-free computation—or signal—σ_s^2

$$NSR = \frac{\sigma_n^2}{\sigma_s^2}. \quad (1)$$

Without loss of generality, we assume unbiased entities, i.e., their mean is zero, and suggest how to satisfy such a requirement.

The structure of the paper is as follows. Section 2 and Section 3 provide the *NSR* at the output of a linear convolution block and a nonlinear function, respectively; altogether, the two results will

- *C. Alippi is with CNR-CESTIA, c/o Dip. Elettronica e Informazione, Politecnico di Milano, P.za L. da Vinci, 32, 20133 Milano, Italy. E-mail: alippi@elet.polimi.it.*
- *L. Briozzo is with SGS-Thomson Microelectronics, Via Olivetti 2, Agrate Brianza, Milano, Italy. E-mail: Luciano.BRIOZZO@st.com.*

Manuscript received 29 Apr. 1997.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 104978.

also be used to deal with nonlinear convolvers. The effect of additive perturbations is briefly tackled in Section 4. Section 5 evaluates the effects of the error generation and propagation along pipelined blocks and considers a feedforward neural network as a synthesis case study. To make the results meaningful and to illustrate the application of this methodology, we focus on digital implementations. In this case, perturbations are caused by data quantization which affects, incrementally, the cascaded steps in the computational chain. An analysis of the methodology based on the use of two case studies is, finally, given in Section 6.

2 The *NSR* at the Output of a Linear Convolution Block

Let us consider a single linear convolver whose computation involves the evaluation of the scalar product x between the n-dimensional column vector of the inputs I and the row vector of the coefficients W

$$x = (W, I) = \sum_{i=1}^{n} W_i I_i = WI. \tag{2}$$

A bias term can be introduced by considering the convolver as fed by a virtual input which always assumes value one. For the sake of clarity, we neglect the bias contribution since it does not substantially modify the analysis.

The error affected device operates on the correspondent perturbed vectors I_p and W_p and provides the perturbed scalar product $x_p = W_p I_p$, where $W = W_p + \delta W$, $I = I_p + \delta I$, and δW and δI are the perturbation vectors. As an example, in digital implementations, perturbations may be caused by truncation or rounding of weights and inputs. Errors δW are local, in the sense that the perturbation affects the coefficients of the specific convolver. Consequently, δI generally accounts both for errors affecting locally the inputs and perturbations generated in previous computations and propagated up to the considered device. The effective perturbation is, therefore, of additive type in both types of perturbation. If the device is the first one in the pipeline, we have a unique perturbation affecting the inputs locally. In the following, we will assume each component of the perturbation vector δI to be a zero mean independent and identically distributed random variable. The error δx between the ideal and the error affected convolution can be expressed as

$$\delta x = x - x_p = (W, I) - (W_p, I_p) = (W_p, \delta I) + (\delta W, I). \tag{3}$$

To compute the NSR at the output of the convolution block, we have to evaluate the variance of the noise $Var[\delta x]$ and that of the signal $Var[x]$. To this end, we consider first the case of a given device for which W, W_p, and δW are fixed entities, i.e., the convolution coefficients have been computed off line and the δW perturbation has been applied. $E[\delta x]$ and $Var[x]$ can be obtained by taking expectations with respect to the I and the δI domains and provide [11]

$$E[\delta x] = (\delta W, M_I), \quad Var[\delta x] = E\left[\left[(W_p, \delta I) + (\delta W, \Psi)\right]^2\right], \tag{4}$$

where M_I is the column vector containing the input means and $\Psi = I - M_I$. From (4), the perturbation δx is biased because of the a priori nonnull M_I. By subtracting M_I from each input vector, we assure, as we assume, $E[\delta x] = 0$. This also has the effect of reducing the word length at the architectural level, thus saving silicon area. Finally, $Var[\delta x]$ becomes

$$Var[\delta x] = WC_{\delta I}W^T + \delta W\left[C_I - C_{\delta I}\right]\delta W^T, \tag{5}$$

where $C_{\delta I}$ and C_I are the covariance matrices of the perturbation on inputs and inputs, respectively. See [11] for the proof.

It is reasonable to assume $C_{\delta I}$ to be diagonal, since the perturbations on inputs are mutually independent; for C_I, this works only if inputs are unrelated. By observing that $C_\Psi = C_I$, we can finally compute the expectation and the variance of the signal

$$E[x] = E[(W, \Psi)] = 0; \quad Var[x] = Var[(W, I)] = WC_I W^T. \tag{6}$$

Finally, the *NSR* for a given linear convolver becomes the ratio between (5) and (6)

$$NSR = \frac{Var[\delta x]}{Var[x]} = \frac{WC_{\delta I}W^T + \delta W\left[C_I - C_{\delta I}\right]\delta W^T}{WC_I W^T}. \tag{7}$$

In digital implementations, it is reasonable to assume that perturbations on inputs are independent (e.g., truncation again) with the same variance $\sigma^2_{\delta I}$. When this holds also for inputs, with each input having the same variance σ^2_I, (7) reduces to

$$NSR = \frac{\sigma^2_{\delta I}}{\sigma^2_I} + \frac{(\delta W, \delta W)}{(W, W)}\left[1 - \frac{\sigma^2_{\delta I}}{\sigma^2_I}\right]. \tag{8}$$

It is simple to extend the validity of (5) from the specific case of a dedicated convolver to a general purpose one, i.e., we move to a device which implements an ensemble of linear convolvers. In this case, weights and perturbations on weights must be modeled as zero mean random variables. Note that we do not require the weights to be independent but only that weights and perturbations on weights are unbiased and independent from inputs and perturbations on inputs. This grants $E[\delta x] = 0$. By taking expectations in (5) with respect to δW and W,

$$Var[\delta x] = tr(C_W C_{\delta I}) + tr\left(C_{\delta W}\left[C_I - C_{\delta I}\right]\right), \tag{9}$$

where tr is the trace operator and C_W and $C_{\delta W}$ are the covariance matrices of weights and perturbation on weights, respectively. In the reasonable case of independent perturbations, $C_{\delta I}$ and $C_{\delta W}$ become diagonal matrices and (9) becomes

$$Var[\delta x] = \sigma^2_{\delta I}\sum_{i=1}^{n}\sigma^2_{W,i} + \sigma^2_{\delta W}\sum_{i=1}^{n}\sigma^2_{I,i} - \sigma^2_{\delta I,i}\sigma^2_{\delta W,i}. \tag{10}$$

As far as the signal is concerned, we can easily obtain that

$$Var[x] = tr(C_W C_I), \tag{11}$$

from which

$$NSR_E = \frac{tr(C_W C_{\delta I}) + tr\left(C_{\delta W}\left[C_I - C_{\delta I}\right]\right)}{tr(C_W C_I)}. \tag{12}$$

If we further assume that W and δW are mutually independent, with the same variance σ^2_W and $\sigma^2_{\delta W}$ for each component of the vectors, it is easy to derive from (10) and (11) that, for the network ensemble case,

$$NSR_E = \frac{\sigma^2_{\delta I}}{\sigma^2_I} + \frac{\sigma^2_{\delta W}}{\sigma^2_W}\left[1 - \frac{\sigma^2_{\delta I}}{\sigma^2_I}\right]. \tag{13}$$

By neglecting the negative contribution, expression (13) becomes formally similar to that suggested in [7], but without assuming any restricting hypotheses.

3 The *NSR* at the Output of a Nonlinear Block

To make the mathematics more amenable, we consider functions $y = f(x), x \in \Re, y \in \Re^2$. Actually, this is not a strict restriction, since the hypothesis holds in most of applications. In some cases, a hard-limited threshold function, such as the Heaviside step function $H(x)$, is applied to the convolution output [2]. $H(x)$ can be seen as the limit of a sigmoidal function $S(x)$ when its temperature T

tends to infinity. More specifically, for a sufficiently large T, $S(Tx) = (1 + e^{-Tx})^{-1}$ is a twice differentiable approximation of $H(x)$. An extension to a multivalued step function is straightforward.

As with the scalar product, we have to determine the stochastic features associated with the noise and the signal by computing means and variances. The error-affected computational flow receives a perturbed scalar x_p and generates a perturbed value y_p. From the definition,

$$E[\delta y] = E[y(x) - y(x_p)] = E[y(x) - y(x - \delta x)], \tag{14}$$

where the expectation is taken with respect to x and δx. Under the reasonable assumption that x and δx are independent random variables (or, more correctly, that the term $E[x\delta x]$ is negligible), a good approximation of $E[\delta y]$is

$$E[\delta y] = -\frac{\sigma_{\delta x}^2}{2} E[y_x''(x)] = -\frac{\sigma_{\delta x}^2}{2} \int y_x''(x)\Omega_x dx. \tag{15}$$

See [11] for the proof. Equation (15) is unknown since, in general, we don't know the true probability density function -*pdf*- Ω_x of x. However, based on the available data, we can, nevertheless, consider the empirical *pdf*

$$\tilde{\Omega}_x(x) = \frac{1}{N} \sum_{i=1}^{N} D_\delta(x - x_i), \tag{16}$$

where $D_\delta(\cdot)$is the Dirac's delta function. This relies on the fact that, in many applications, the number of data N used to configure the convolver's coefficients is generally large in order to grant a good parameter estimation [2]. As a direct consequence of the law of large numbers, $\tilde{\Omega}_x$ converges weakly to Ω_x. Therefore, for a sufficiently large N, $\tilde{\Omega}_x$ can be substituted with Ω_x and (15) becomes

$$E[\delta y] \cong -\frac{\sigma_{\delta x}^2}{2N} \sum_{i=1}^{N} y_x''(x_i). \tag{17}$$

We can now consider $Var[\delta y]$. By leaving details to [11], we obtain that

$$Var[\delta y] = E[y_x'(x)^2]\sigma_{\delta x}^2 + \frac{\sigma_{\delta x}^4}{4}\left(3E[y_x''(x)^2] - E[y_x''(x)]^2\right). \tag{18}$$

As with the mean, we estimate the variance by considering $\tilde{\Omega}_x$

$$Var[\delta y] = \frac{\sigma_{\delta x}^2}{N} \sum_{i=1}^{N} y_x'(x_i)^2 + \frac{\sigma_{\delta x}^4}{4}\left(\frac{3}{N} \sum_{i=1}^{N} y_x''(x_i)^2 - \left[\frac{1}{N} \sum_{i=1}^{N} y_x''(x_i)\right]^2\right). \tag{19}$$

Since we do not assume a large fan-in for the convolver, as in [7], [8], we cannot compute the mean and the variance for the signal x in a close form. Nevertheless, since we know $\tilde{\Omega}_x$, we have that

$$E[y] \cong \mu_y = \frac{1}{N} \sum_{i=1}^{N} y(x_i) \qquad Var[y] = \sigma_y^2 \cong \frac{1}{N} \sum_{i=1}^{N} y^2(x_i) - \mu_y^2. \tag{20}$$

The *NSR* at the output of the nonlinear block finally becomes the ratio of (19) to (20).

4 The *NSR* as a Consequence of an Additive Perturbation

In digital realizations, the output of a computational block is often truncated or rounded before performing subsequent computations. In other cases, an additional perturbation may affect the outputs. To model such actions, we consider a module which inputs a x value and outputs a perturbed value x_p such that $x = x_p + \delta x$, where $\delta x$is a random variable with zero mean and $\sigma_{\delta x}^2$ variance. Trivially, if σ_x^2 is the variance of the signal, the *NSR* directly comes from its definition (1).

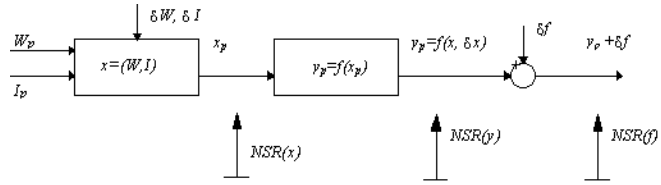


Fig. 1. A neuron, a simple pipelined computational block.

5 The NSR at the Output of Pipelined Digital Blocks

In order to illustrate how previous results can be applied in a real computational chain, we consider in this section the implementation of a simple algorithm in digital hardware. The perturbations affecting the computation are due to quantization of the signals involved. Quantization, which implies a reduction of the signal resolution, can be applied to inputs, convolution coefficients, or, at the end of intermediate computational steps, on partial results. Another source of quantization error comes from the discretization of continuous functions, such as those utilized in nonlinear computational steps. Quantization can be obtained by considering truncation, rounding or jamming techniques (see [7] for a review). As with [8] and [9], we assume that errors coming from rounding and truncation are discrete random variables subject to uniform distributions. Such errors are independent from each other and from all inputs and outputs. As pointed out in [9], a preliminary step requires investigation of the minimum and the maximum values assumed by the weights; afterward, a subsequent rescaling phase is generally envisaged for resolution efficiency. We definitely confine ourselves to this case.

The quantization error, obtained by removing from a k bits number the q less relevant ones, has approximately zero mean and variance $\sigma_T^2 = 2^{2q}/3$ for truncation and $\sigma_R^2 = 2^{2q}/12$ for rounding.

As an example, we consider the case of a general purpose neuron that is to be implemented in VLSI hardware (possibly within a network of neurons) and which must be configured to receive $n = 7$ real weights, each defined within the [–7, 7] interval and able to process uniformly distributed inputs in the [–3, 3] interval. A two's complement fixed point representation is considered; for simplicity, we assume the integer parts to be represented without errors and that the decimal part is characterized by b_I and b_W bits to represent inputs and weights, respectively. Truncation is considered.

Since we are configuring a general purpose neuron, it is also reasonable to assume that the weights are uniformly distributed in their definition interval. Note that we can deal directly with (18). We assume, as with [9], the hyperbolic tangent activation function to be uniformly defined in the [–1, 1] interval. Different models can be obviously considered, e.g., the one suggested again in [9]. By recalling that, for a hyperbolic tangent function $y' = (1 - y^2)$ and $y'' = -2y'y$, we can easily compute the expectations needed in (18) and (20): $E[y'^2] = 8/15$, $E[y''^2] = 32/105$, $E[y''] = 0$, $E[y] = 0$, $E[y^2] = 1/3$. With respect to Fig. 1, the NSR_E at the neuron output is

$$NSR_E(f) = NSR_E(y) + \frac{\sigma_{\delta f}^2}{\sigma_y^2}. \tag{21}$$

The perturbation δ_f might be seen as a truncation or induced by a look-up table realization of the nonlinear activation function. We indicate with b_f the number of bits, yet to be defined, necessary to store in the look-up table the output values. Nothing changes, however, if the perturbation δ_f, instead of being introduced by a look-up table, is derived from a device implementing

the nonlinear function, e.g., see [12]. From (18) and (21) we obtain, that [11]

$$NSR_E(f) = 3\sigma_{\delta f}^2 + \frac{8}{5}\sigma_{\delta x}^2 + \frac{24}{35}\sigma_{\delta x}^4. \quad (22)$$

From the biquadratic equation (22), a necessary condition for a feasible solution is $\sigma_{\delta f}^2 < NSR_E(f)/3$. If we tolerate, at most, an $NSR_E(f)$ loss in accuracy at the neuron output, we have, from (22),

$$\sigma_{\delta x}^2 < 35\Big/24\Big(-28 + \sqrt{784 - 840(3\sigma_{\delta f}^2 - NSR_E(f))}\Big).$$

For simplicity, let us assume that there are no errors at the accumulator level, i.e., that the errors affecting the scalar product have been generated only by propagation of errors on *W* and *I*. If this is not the case, then we have to consider an additional additive perturbation on the accumulator as we did for δ_f. Under the above mentioned hypotheses, from (9)

$$\sigma_{\delta x}^2 = n\sigma_W^2\sigma_{\delta I}^2 + n\Big(\sigma_I^2 - \sigma_{\delta I}^2\Big)\sigma_{\delta W}^2 \cong \frac{7}{9}(49\ 2^{-2b_I} + 9\ 2^{-2b_W}). \quad (23)$$

Finally, the relationship between quantization on weights, inputs, outputs and loss in accuracy is

$$y1 = \frac{7}{9}(49\ 2^{-2b_I} + 9\ 2^{-2b_W}) < 35\Big/24\Big(-28 + \sqrt{784 - 840(3\sigma_{\delta f}^2 - NSR_E(f))}\Big) = y2. \quad (24)$$

Each point $P\big(b_I, b_W, b_y, NSR_E(f)\big)$ satisfying (24) generates a loss in accuracy at the neuron output that is smaller than $NSR_E(f)$. If we are considering the synthesis phase, i.e., we wish to synthesize the architecture at the register level, the tolerated loss in accuracy at the neuron output $NSR_E(f)$ is given and we have to determine how many bits we need to represent the inputs, the weights and the output. Conversely, in the analysis phase, the perturbations are given, i.e., the number of bits used to represent values are set and we want to measure the resulting loss in accuracy at the device's output.

An example of synthesis is given in Fig. 2 for the case $NSR_E(f) = 0.1$ (we tolerate a 10 percent loss in accuracy according to *NSR*). The numbers of bits used to represent the decimal part of the inputs and outputs are given on the abscissa axis; the model family $y1$ (which is parameterized in b_W) and $y2$ are also given. Graphically, the points satisfying (24) are those for which the condition $y1 < y2$ is satisfied. In Fig. 2, we can immediately identify two interesting solutions: $P1\big(b_f = b_I = 2, b_W = 4\big)$ and $P2\big(b_f = b_I = b_W = 3\big)$. P1 is of particular interest if the look-up table plays the most relevant role in occupying silicon area (i.e., we want to minimize the number of bits used to represent the inputs). Conversely, P2 is to be preferred if the number of weights is dominant in occupying silicon area (i.e., we wish to minimize the number of bits used to represent the weights).

The extension from a single neuron example to a whole network is directly realizable. Equation (24) has been obtained from (23) by assuming that the neuron's inputs are mutually independent; if this not the case, then we simply have to consider (10) instead of (23) and to adapt it. The unique difference is in $y1$, since now the weights and inputs assume different covariance matrices for each neural layer. For a whole network, it is reasonable to consider a unique variance for all weights; the main difference is therefore in the inputs and the perturbations on inputs covariances. We can, thus, easily dimension inputs, internal registers, and outputs by graphically dimensioning a neuron for each layer as we have shown in this section.

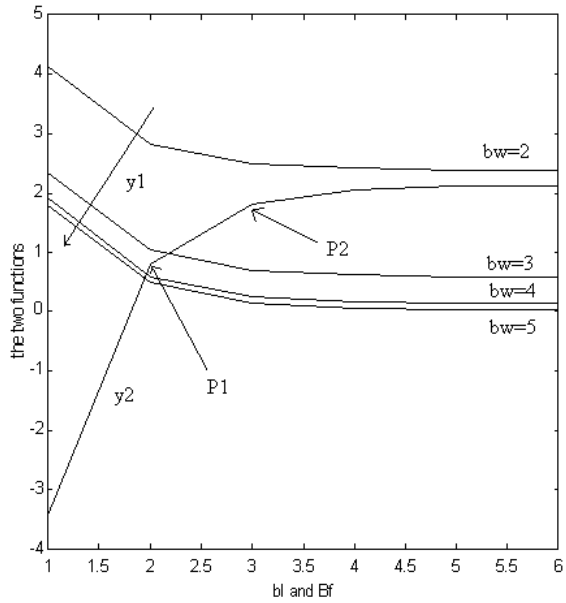


Fig. 2. The synthesis phase for a neuron: the $NSR_E(f) = 0.1$ case .

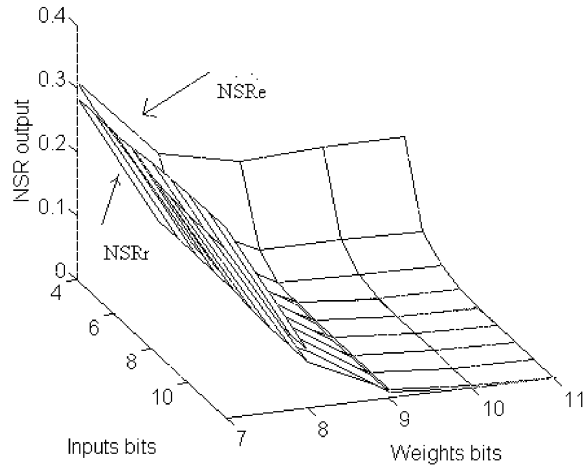


Fig. 3.No errors at the accumulator level.

6 Experimental Results

6.1 Case Study 1: The Sensitivity Analysis in a Digital Architecture for Signal Processing

The experiment deals with function approximation and, in particular, with the configuration of a feedforward network trained to approximate a $\Re \to \Re$ function defined in the [–5, 5] interval. The best neural network possesses five hidden neurons: This is an example of a reduced fan-in case.

Furthermore, since hidden neurons process the same input values, the inputs of the output neuron are strongly dependent. We assume all weights in the network to be represented with the same resolution. The goal is to evaluate the *NSR* as predicted by the theory at different points of the architecture and to compare it with the experimental derived counterpart. Errors are due to truncation of neural values involved in the computation. In this experiment, finite precision affects the inputs and the weights. No further errors affect the accumulator which is used to store the resulting convolution value. From the results given in Fig. 3; we can see that the *NSRe* estimated by the theory at the output of the output neuron nicely approximates the experimental one *NSR*r. As expected, by increasing the resolution, *NSR* tends to zero, i.e., the implemented

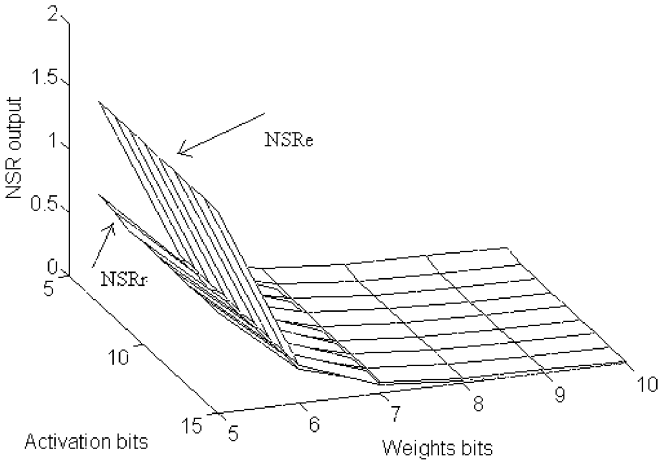


Fig. 4. One bit to represent the decimal part of inputs.

algorithm tends to the ideal one. On the other hand, if we consider less then nine bits to represent the weights or less than five bits for the inputs, the loss in accuracy drastically increases. In our case, since a two's complement representation has been considered, a minimum of five bits for the inputs means that, in a fixed point notation, we need a bit to represent the decimal part.

As a second experiment, we considered an architecture in which the number of bits used to represent the decimal part of inputs was fixed to one. Perturbations also affect the weights and the accumulator output, i.e., there is an additive error caused by truncation at the network's output. The situation is shown in Fig. 4. We can see *NSRe* overestimates the real one *NSRr*; the approximation is satisfactory provided that the perturbation affecting the scalar product is small enough to guarantee the validity of (19). We should note that *NSRe* becomes a bad estimate of *NSRr* only when the perturbation almost fully corrupts the signal

6.2 Case Study 2: The Sensitivity Analysis in a Digital Architecture for Image Processing

This application refers to the development of a dedicated digital VLSI hardware to detect the presence of linear defects in images. The portion of the architecture with the highest computational load can be logically subdivided into two cascaded modules: a feature extractor module followed by a decision module to classify the presence/absence of linear defects in the object [13]. The first module is a constrained convolver whose 9 × 5 weights mask has been trained with defect/no-defect examples. The second block, or defect identifier, can be implemented with a simple decision algorithm based on the convolution values. However, critical situations may require finer decisions to be implemented outside the chip. Here, we will focus our attention on the neuron-convolver processing module, which receives the input data and provides the convolution result to the chip's output.

After experimental analyses, it was decided to reduce the input resolution from eight to six bits. The scalar product was constructed by placing the partial products in a look-up-table. No errors were introduced at the accumulator level and the outputs, characterized by an eight-bit resolution, were represented with six bits. All data reduction was implemented with truncation. The final architecture, developed at SGS-Thomson, together with the specified perturbations, can be *functionally* represented, as shown in Fig. 5. To validate the architectural choices, we have to compute the *NSR* at the output of the convolver. According to the data flow of Fig. 5, the *NSR(f)* at the convolution output is simply the ratio of the variances of the perturbations propagated along the computational chain to that of the signal, namely the convolution output. If

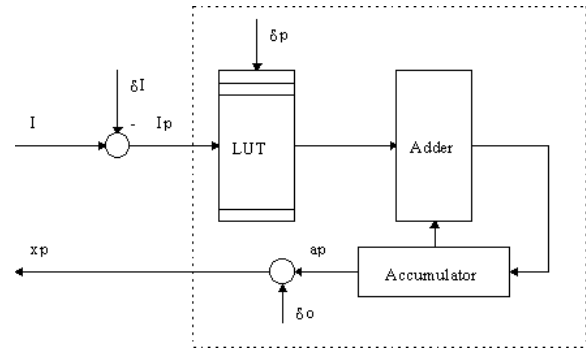
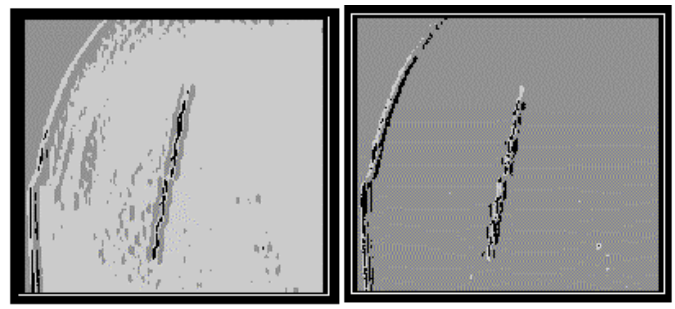


Fig. 5. The functional description of the architecture.



(a) (b)

Fig. 6. (a) the ideal convolution b: the real convolution, (b) the real convolution.

we indicate with $\sigma^2_{\delta o}$ and $\sigma^2_{\delta p}$ the variances of the perturbations, δo caused by truncating the scalar product output and δp associated with the resolution of the partial product stored in the LUT

$$NSR(ap) = \frac{n\sigma^2_{\delta p}}{\sigma^2_x} + \frac{\sigma^2_{\delta I}}{\sigma^2_x}; \qquad NSR(f) = NSR(ap) + \frac{\sigma^2_{\delta o}}{\sigma^2_x}.$$

Finally, from (5), we obtain

$$NSR(f) = \frac{WC_{\delta I}W^T}{\sigma^2_x} + \frac{n\sigma^2_{\delta p}}{\sigma^2_x} + \frac{\sigma^2_{\delta o}}{\sigma^2_x} = \frac{\sigma^2_{\delta I}|W|^2 + n\sigma^2_{\delta p} + \sigma^2_{\delta o}}{\sigma^2_x}.$$

The signal variance has been evaluated according to (20) and provided 2380, and the squared magnitude of the weights vector is 0.43. Since all truncations introduce a two bits reduction, the final noise to signal ratio is 0.078 which provides a good estimate of the measured one *NSR* = 0.075. The loss in accuracy introduced by the considered architectural design is, therefore, of 7.8 percent.

An example of accuracy degradation is given in Fig. 6, where a portion of a convoluted image is given. Fig. 6a presents the image which has been convoluted with the error-free device, while Fig. 6b presents the image generated by the real convolver. We can see that the real device introduces a visible loss in accuracy; this loss can be promptly estimated by the suggested methodology.

As a final remark, we must mention that the implemented device is intended only as a prototype; register dimensioning was, in fact, empirically determined and tailored to a limited set of images. On this set, the device behaves sufficiently well. The methodology developed in this paper, however, presents a critical value of *NSR* which suggests that a more conservative register dimensioning would allow a more robust implementation of the device.

7 Conclusions

In this paper, we have presented a sensitivity analysis to deal with algorithms containing scalar product evaluations and nonlinear function computation. The stochastic framework investigates the

effects caused by perturbations at different levels of the computational chain by estimating the induced loss in accuracy on the basis of a noise to signal figure of merit. Results have been tailored to digital implementations where the main cause of noise is associated with quantization effects. Extensions to others operators, e.g., max and modulus, are currently under study.

REFERENCES

[1] The Fermi Group, "A Digital Front-End and Readout Microsystem for Calorimetry at LHC: The FERMI Project," *IEEE Trans. Nuclear Science*, vol. 40, no. 4, pp. 516-530, Aug. 1993.

[2] W.K. Pratt, *Digital Image Processing*. Wiley Interscience, 1978.

[3] J. Hertz, A. Krogh, and R.G. Palmer, *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

[4] S.G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 2, no. 7, pp. 674-693, July 1989.

[5] M. Stevenson, R. Winter, and B. Widrow, "Sensitivity of Feedforward Neural Networks to Weights Errors," *IEEE Trans. Neural Networks*, vol. 1, no 1, Mar. 1990.

[6] C. Alippi, V. Piuri, and M. Sami, "Sensitivity to Errors in Artificial Neural Networks: A Behavioural Approach," *IEEE Trans. Circuits and Systems-I*, vol. 42, no 6, June 1995.

[7] S. Piché, "The Selection of Weights Accuracies for Madalines," *IEEE Trans. Neural Networks*, vol. 6, no. 2, Mar. 1995.

[8] J. Holt and J. Hwang, "Finite Precision Error Analysis of Neural Network Hardware Implementations," *IEEE Trans. Computers*, vol. 42, no. 3, Mar. 1993.

[9] G. Dundar and K. Rose, "The Efects of Quantization on Multilayer Neural Networks," *IEEE Trans. Neural Networks*, vol. 6, no. 6, Nov. 1995.

[10] B. Hassibi and D.G. Stork, "Second Order Derivative for Network Pruning: Optimal Brain Surgeon," *Proc. NIPS5*, 1993.

[11] C. Alippi and L.Briozzo, "Accuracy vs. Precision in Digital VLSI Architectures for Signal Processing," Internal Report CNR-CESTIA 97-02, 1997.

[12] C. Alippi and G. Storti-Gajani, "Simple Approximation of Sigmoidal Function: Realistic Design of Digital Neural Networks Capable of Learning," *Proc. IEEE-ISCAS*, Singapore, 11-14 June 1991.

[13] M. Valle, D. Caviglia, M. Cornero, G. Nateri, and L. Briozzo, "A VHDL Based Design Methodology the Design Experience of a High Performance ASIC Chip," *Proc. Euro—VHDL Conf.*, Grenoble, France, 19-23 Sept. 1994.