Efficient Training of Neural Gas Vector Quantizers with Analog Circuit Implementation

Stefano Rovetta, *Member, IEEE,* and Rodolfo Zunino, *Member, IEEE*

***Abstract*—This paper presents an algorithm for training vector quantizers with an improved version of the Neural Gas model, and its implementation in analog circuitry. Theoretical properties of the algorithm are proven that clarify the performance of the method in terms of quantization quality, and motivate design aspects of the hardware implementation. The architecture for vector quantization training includes two chips, one for Euclidean distance computation, the other for programmable sorting of codevectors. Experimental results obtained in a real application (image coding) support both the algorithm's effectiveness and the hardware performance, which can speed up the training process by up to two orders of magnitude.**

I. Introduction

THE LITERATURE provides a large amount of training algorithms for vector quantization (VQ) [1] systems: conventional [2] and neural-network [3]–[9] algorithms witness the interest in VQ as an effective paradigm for domain analysis, signal processing, and information coding. As to hardware implementations of VQ, most efforts have been spent in supporting the winner-take-all (WTA) function in both analog and digital technology [10]–[16]. Other approaches optimize codebook search for real-time applications [17]. Visual information coding [18] is usually the target application. VQ performs effectively at very low bit rates, and its notable computational cost justifies the effort for hardware implementations.

Comparatively, less work seems to have been done on hardware support for VQ training algorithms, despite codebook training is usually the computation-intensive part in a VQ system setup. In addition to the direct modeling of Kohonen's Self-Organizing Maps (SOM's) [19], [20], very large scale integration (VLSI) training architectures have been described in [21], [22] and, more recently, [23]–[25]. The complexity of mapping theory into architectures is possibly the major issue hindering the hardware (HW) support of training algorithms. A dynamic adjustment of codevectors can involve a wide-spread circulation of information, thus demanding complex wiring and coordination. It is, therefore, no surprise that the latest trainable encoder implements a nearest-neighbor model [24], which does not require any interprototype connectivity.

This paper tackles the problem of HW support for VQ training in high-dimensional domains requiring huge computational cost (typically video coding). The proposed VQ training algorithm performs effectively in terms of distortion noise, and has HW-amenable features; hence, the related implementation in VLSI circuitry is also illustrated. Codevector positioning is driven by an improved version of the Neural Gas (NG) model [4], whose simple topology makes it possible to minimize interneuron connectivity.

This latter property makes NG appealing with respect to other VQ-training methods such as SOM's [5], which involve a fixed grid of neuron interconnections and can notably complicate HW implementation. Moreover, the NG model seems computationally more efficient than affine topology-free approaches [8], [9], as the latter ones require global computation of frequencies and probabilities, and may therefore pay an increased computational cost. As far as representation quality is concerned, the performance of NG training has also been proved experimentally [4] to overcome standard VQ algorithms in the literature, such as SOM's or the k-means algorithm [2].

Previous research showed that NG can yield impressive efficiency even on conventional architectures [26], and that a plastic algorithm [17] ensures convergence and can minimize codebook size. NG is appealing as prototypes mostly operate independently of one another; the only information-exchanging step of the algorithm involves codevector sorting (DSD).

This paper improves the NG algorithm by giving both analytical and computational support for partial sorting, which speeds up computation and simplifies the HW, especially in the presence of large codebooks. The circuitry implements the critical steps of VQ training, namely, distance computation (DCD) and DSD. The design strategy integrates multiple chips; splitting the overall schema into components simplifies the designer's configuration. This connects the present algorithm to previous work, which led to an analog chip for full-search codebook scanning [27]; moreover, a circuit for sorting was drafted in [28].

The result of the presented research is a novel framework for effective VQ training, where theory drives HW implementation. Experimental verifications include: 1) tests to validate theoretical expectations and 2) measurements of the performance of the supporting HW architecture. Empirical evidence shows that the approach can reduce training time by up to two orders of magnitude, without affecting performance quality. Section II presents the theoretical framework, comprehending the algorithm formulation and the theory for partial sorting. Section III describes the implementation circuitry, and

Manuscript received September 9, 1998; revised February 26, 1999. This work was supported by the Italian Ministry for University of Scientific and Technological Research. This paper was recommended by Associate Editor B. Linares-Barranco.

The authors are with the Department of Biophysical and Electronic Engineering (DIBE), University of Genoa, 16145 Genova, Italy.

Publisher Item Identifier S 1057-7130(99)04877-6.

Section IV reports experimental results. Concluding remarks are made in Section V.

II. Theoretical Aspects of VQ Training

A. The VQ-Training Algorithm

The VQ training algorithm is based on the NG model for neural network training. NG places a set of codevectors in a data space according to the empirical distribution of a training set, and aims to minimize average distortion. Theory shows that the asymptotical positions of codevectors span a uniform coverage over training data [4]. Empirical evidence indicates that, on average, NG results in a smaller distortion, as compared with other VQ algorithms; moreover, convergence typically requires fewer iterations.

The model does not assume any topology for neuron connectivity. During training, codevectors are sorted in terms of their Euclidean distances from input samples; each codeword is adjusted according to its rank. Codewords are denoted by $\{\mathbf{w}_n, n = 1, \cdots, N\}$ and lie in a d-dimensional domain space, X. The algorithm can be outlined as follows.

For iterations $t = 0$ to T:

1) *draw* a training sample, $\mathbf{x} \in X$;
2) $\forall$ codevector, w_n, *compute* the Euclidean distance
$$d_n = \|\mathbf{w}_n - \mathbf{x}\|, \quad n = 1, \cdots, N \tag{1}$$
3) *partial sort*: extract the first K codewords in order of increasing $d_n (K \ll N)$;
let $n(k)$ denote the index of the kth codeword on the sorted list;
4) *adjust* the subset of codevectors according to their ranks on the partial list
$$\mathbf{w}_{n(k)}^{(t+1)} = \mathbf{w}_{n(k)}^{(t)} + \varepsilon(k,t) \cdot \left(\mathbf{x} - \mathbf{w}_{n(k)}^{(t)}\right), \quad k = 0, \cdots, K. \tag{2}$$

The rewarding function ε supports a decreasing learning rate and balances the distribution of weight updates during training. An implementation of the scheduling functions is given in [4]

$$\varepsilon(k,t) = \eta_0 (\eta_T/\eta_0)^{t/T} \exp\left(-\frac{k}{\lambda(t)}\right)$$
$$\lambda(t) = \lambda_0 \left(\frac{\lambda_T}{\lambda_0}\right)^{t/T}, \quad \eta_0 > \eta_T > 0; \quad \lambda_0 > \lambda_T > 0. \tag{3}$$

According to (3), training evolves from a distributed-activation pattern, in which several codevectors are adjusted at the same time, to a true WTA schema, in which only one codeword is affected by a training sample. The NG model is computationally interesting, as most of a codevector's activity proceeds locally: 1) DCD and 2) weight adjustments exploit local information, thus enabling the use of codeword-embedded circuitry [27]. Sorting is the only process requiring a network-wide circulation of information to evaluate list ranks k.

When considering the computational costs of the sorting operation, one could think of other VQ algorithms of the type (2) which update all codewords every training cycle, and where ε is not a function of the rank k, but of the distance d_n itself. This would remove the need for WTA circuitry and codeword sorting altogether. In fact, sorting in codeword scoring provides the training strategy with mechanisms related to robust statistics. This mainly deals with the practical well-known problem of dead vectors: if weight adjustment depends on d_n, the risk is high that the annealing mechanism implied by (3) fails to recover "distant" codevectors that might have been moved away occasionally or badly initialized. Making codevector displacements dependent on their ranks ensures that even the farthest codewords are significantly influenced by every training sample. The theoretical justification of sorting finds a thorough confirmation in experimental practice [4], [17], showing the extremely low rate of dead vectors derived from an NG training, as compared with other methods.

However, a downright implementation of the original NG algorithm would imply complete sorting and network-wide communications. In large codebooks for real applications, however, a complete sorting would complicate connectivity and layout design [12], [29]. This motivates partial sorting from an architectural perspective; the theoretical validity of the approach is proved in the following.

B. Theoretical Support for Partial Sorting

Considering a subset of the sorted list should not affect training results, as compared with those yielded by the original NG. This is possible as NG implements stochastic optimization, whose progress can be modeled by two additive quantities, i.e., cost and noise. The former depends on the distortion brought about by quantization, the latter results from random fluctuations during the minimum-search process.

The theoretical baseline for the presented algorithm is fairly simple: for a complete sorting, the gain in distortion for the last codevectors on the list becomes comparable with the loss due to random fluctuations. As a result, a complete sorting is practically insignificant to optimization. To prove this property, the sensitivity of training to the sorting process is analyzed and a novel interpretation of the codevector-adjustment strategy is provided.

In order to simplify the analysis and without loss of generality, one can assume a continuous-time training process (involving an infinite number of training samples) and a continuous-size codebook (reflecting a very large number of codevectors). Actually, such idealization is a good model of practically interesting problems; that is, VQ training in high-dimensional applications with large codebooks (e.g., image coding). Under these assumptions, normalized sensitivity derives from the weight-update rule (3), and is formally defined as

$$S(k,t) = -\frac{\lambda_0}{\eta_0}\frac{\partial}{\partial k}\varepsilon(k,t) = \left(\frac{\eta_T \lambda_0}{\eta_0 \lambda_T}\right)^{t/T} \exp\left(\frac{-k}{\lambda(t)}\right) \tag{4}$$

where the normalization factor $-\lambda_0/\eta_0$ has been introduced, without loss of generality, to simplify notation in the following math derivations.

When regarded as a function of time for a fixed rank k_0, sensitivity shows how the importance of rank k_0 evolves during

training. The properties of sensitivity can clarify the overall operation of the VQ training algorithm. The sensitivity for the highest rank $k = 0$ (which relates to the best-matching neuron) increases exponentially during training; therefore, identifying the winning neuron becomes more and more important as training progresses. Conversely, the sensitivity for any other rank decreases, hence the importance of each nonwinning position decreases, too.

This behavior matches the originally described features of the NG algorithm [4], and derives from the shape of the rewarding function $\lambda(t)$. From a practical perspective, one should consider that the algorithm's robustness relates strictly to the accuracy of the sorting process. Thus, the robustness of partial sorting increases in time, as can be easily proven

$$\lim_{t\to\infty} S(0,t) = \infty; \quad \lim_{t\to\infty} S(k,t) = 0 \quad \forall k \geq 1. \tag{5}$$

The first limit holds if $\eta_T\lambda_0/\eta_0\lambda_T > 1$, meaning that the top rank becomes more and more important as training progresses. In theory, one might also set the annealing process in such a way that the importance decreases for $k = 0$; previous research [4] and experimental practice indicate that privileging the winner position strongly improves training quality. The second limit is always true and confirms the decreasing importance of minor list ranks while training progresses. The importance $I(k)$ of the kth list position throughout the entire training process is measured by

$$I(k) = \int_0^T S(k,t)\,dt. \tag{6}$$

The integrals (6) must be worked out numerically. Importance becomes insignificant for large values of k; more importantly, a few top positions convey most of the overall importance. This behavior is justified analytically by the following Theorems.

Theorem 1: (Importance decreases with list rank)

$$\forall k \geq 0, \quad I(k) > I(k+1). \tag{7}$$

Proof: The condition $k \geq 0$ reflects the meaning of k as a rank on a list. The inequality can be easily proved by rewriting importance:

$$\begin{aligned} I(k) &= \int_0^T \exp\left(At - \frac{k}{\lambda_0}e^{Bt}\right) dt \\ A &= \frac{1}{T}\ln\frac{\eta_T\lambda_0}{\eta_0\lambda_T} \\ B &= \frac{1}{T}\ln\frac{\lambda_0}{\lambda_T}. \end{aligned} \tag{8}$$

If one uses (8) to reformulate (7), after simple rearrangement, condition (7) can be rewritten as

$$\int_0^T \exp\left(At - \frac{k}{\lambda_0}e^{Bt}\right)\left[1 - \exp\left(-\frac{1}{\lambda_0}e^{Bt}\right)\right] dt > 0 \quad \forall k > 0. \tag{9}$$

The integrand function is positive if

$$\exp\left(-\frac{1}{\lambda_0}e^{Bt}\right) < 1 \Leftrightarrow -\frac{1}{\lambda_0}e^{Bt} < 0$$

which is always true because $\lambda_0 > 0$. □

Theorem 2: The series $I(k)$ tends to zero exponentially

$$\lim_{k\to\infty} I(k) = 0. \tag{10}$$

Proof: Clearly, $I(k) > 0$; one can prove convergence by upper bounding the integrand function

$$\begin{aligned} I(k) &< \int_0^T \exp\left[At - \frac{k}{\lambda_0}\left(1 + Bt + \frac{(Bt)^2}{2}\right)\right] dt \\ &= e^{-k/\lambda_0}\int_0^T \exp\left[(A - kB/\lambda_0)t - \frac{kB^2}{2\lambda_0}t^2\right] dt \\ &= e^{-kB}g(k), \quad g(k) < \infty \quad \forall k > 0. \end{aligned} \tag{11}$$

Inequality (11) implies that the series $I(k)$ is bounded within two series converging to zero, hence

$$\lim_{k\to\infty} e^{-kB}g(k) = 0 \Rightarrow \lim_{k\to\infty} I(k) = 0.$$

Since the bounding series tends to zero exponentially, the series $I(k)$ itself must converge to zero with the same rate, which completes the proof. □

Theorem 3: (Convergence of total importance)

$$\sum_{k=1}^{\infty} I(k) = I_{\mathrm{TOT}} < \infty. \tag{12}$$

Proof: This property is a corollary of the previous one, as the exponential convergence rate of $I(k)$ is a sufficient condition for the convergence of the sum in the assertion. □

Theorem 3 represents the actual theoretical support for partial sorting. One can set a list depth $K < N$ and limit the sorting process to the initial K positions, with an arbitrarily small effect on training results. Remarkably, K can be set once and for all as property (12) is independent of the specific quantization problem. This holds because the theorems follow from the implementations of the scheduling functions (3).

Sample results for a few top ranks are given in Fig. 1, showing sensitivity curves (a) and relative importance values, defined as $i_k = I(k)/I_{\mathrm{TOT}}$ (b). The graph confirms the exponential rate of convergence of the series. The sensitivity graph in Fig. 1(a) witnesses the different importance trend for the top list position $(k = 0)$ with respect to the others: identifying correctly the winning neuron is much more important than any other position on the list. This might indirectly motivate the success of most WTA-based VQ training algorithms such as k means; conversely, taking into account the residual importance justifies the superior performance of NG in representation quality mean-square error (MSE). More importantly, as a positive support for partial sorting, the cumulative distribution of i_k points out that more than 95% of the total importance is conveyed by the 15 top ranks on the list.

III. Circuit Implementation of VQ Training

A. Overall Architecture

The architecture described in this paper uses dedicated circuitry to perform the computation-intensive steps of the training algorithm. The narrow dynamic range of the involved

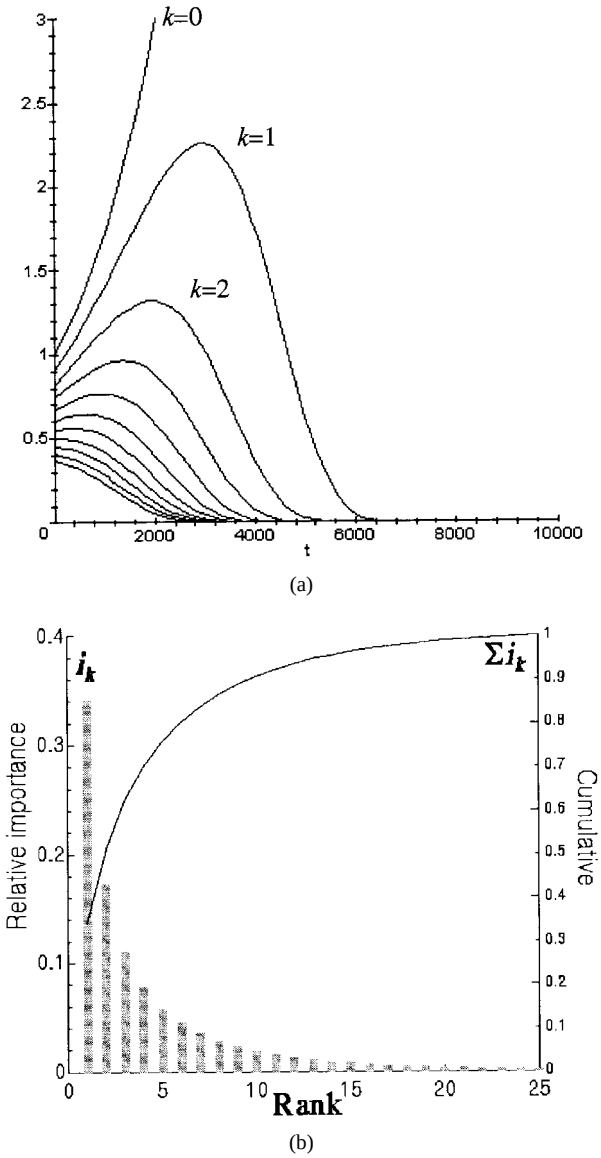


Fig. 1. Sensitivity analysis of the VQ training process. (a) Normalized sensitivities versus time $(k > 0)$. (b) Relative importance i_k versus rank.

quantities (typically less than 10 bits) allows analog implementations; at the same time, by analog circuitry, the architecture attains maximum parallelism in a limited silicon area, thus effectively supporting the large bandwidth required by VQ [30], [31].

The training architecture is sketched in Fig. 2 and implements the algorithm's steps 1) and 2), involving DCD and DSD, respectively. A full HW implementation of the algorithm is feasible, as step 3) is performed locally at the codevector level, too. In the present setup, however, conventional digital processors accomplish this task, for both a structural and practical reason. First, this choice makes it possible to modify the codevector-rewarding function $\varepsilon(k,t)$. Secondly, weight update is computationally much lighter than the previous steps; by contrast, an accurate HW implementation of the exponential functions may prove quite complicated.

The choice of using separate devices for DCD and DSD aims at a practical goal as well. Sorting is only required

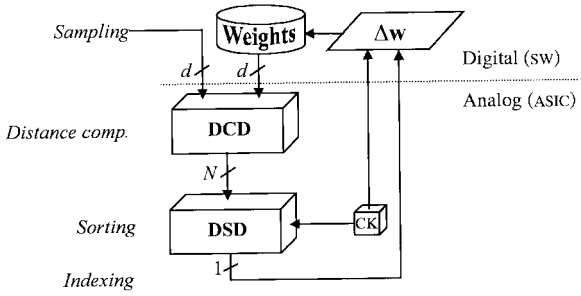


Fig. 2. The VQ training system architecture.

during training, whereas a distance-computation device can also be used at run time for information coding with trained codebooks. The two devices will therefore be considered separately.

B. DCD

Computing the distance of each codevector from the current-input vector is a time-consuming task. Many solutions have been proposed, based on both HW accelerators and on algorithm optimization. In the relevant literature, fully digital approaches are usually employed for distance-computation circuits [32]. Analog realizations can be found especially in the neural network field [33], [34]. Optimizing DCD from an algorithmic point of view, on the other hand, is useful in the case of a single winner. In this situations, many techniques can be applied [32], [35]. Such techniques aim to reduce codebook search time.

The solution proposed here relies on a complete parallelization, in terms of DCD (all vector components enter the computation in parallel) and codebook access (all codevectors are accessed simultaneously). This architectural setup is consistent with several approaches described in the literature [25], [31]. Electronic implementation of a completely parallel architecture is possible, with a reasonable area efficiency, only if one resorts to analog circuitry.

In the presented research, a DCD with the above features has been suitably designed and fabricated. A detailed description of the internal chip architecture [27] is beyond the scope of this paper, which mainly focuses on the overall architectural aspects of the VQ training algorithm. The DCD processes analog vectors (represented as an array of input voltages), stores the codebook in local analog memories (capacitors) and in an external digital memory, and outputs three types of information: the index of the best-matching codevector, encoded as a digital word; the distance value of the best-matching codevector; and the distance values of all the codevectors.

This enables the circuit to act as both the required distance-computing device and a stand-alone VQ encoder, while providing remarkable flexibility for a modular connection of many such circuits. The chip is illustrated by the block diagram in Fig. 3 (a photograph is presented in Fig. 4). The circuit accepts input voltages in the range 0–1 V. The best-matching codevector is evaluated in terms of squared Euclidean distance, expressed as $d_i = \sum_j (\mathbf{w}_{i,j} - \mathbf{x}_j)^2$. Each term of this summation (the output of a squaring block) is a current signal;

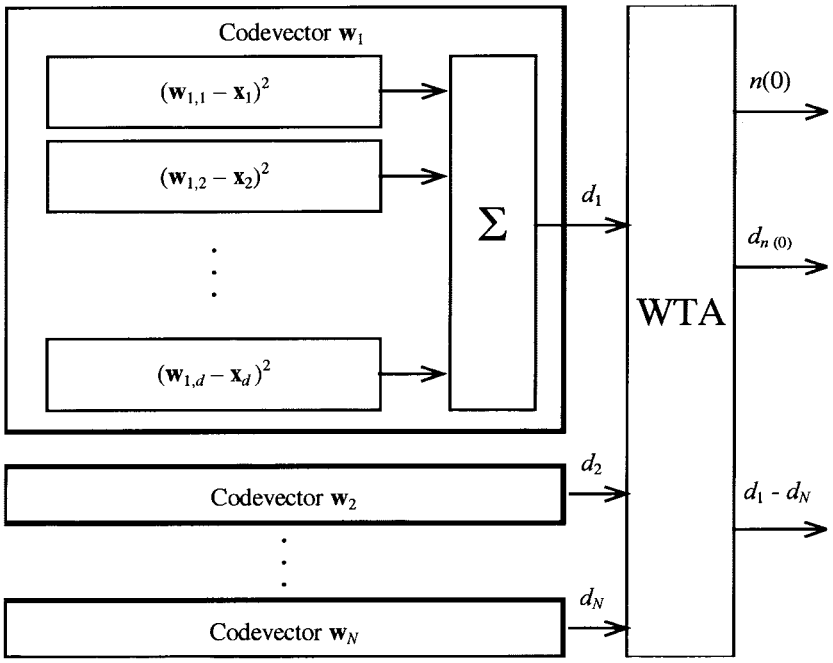


Fig. 3. Block diagram of the DCD.

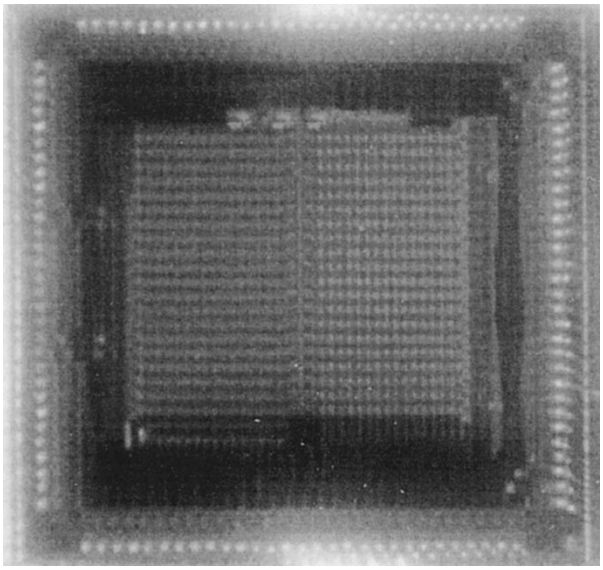


Fig. 4. Chip photograph of the DCD.

hence, the summation itself is greatly simplified. Then, the distance is input to the WTA block, a modified and enhanced version of the well-known circuit designed by Lazzaro *et al.* [13]. This block produces the output signals described above. Other components of the circuit are the refresh circuitry, based on external D/A converters, and the polarization/reference subsystem, implemented with the resistive interpolation biasing technique [36].

C. DSD

The literature offers a variety of HW circuits for sorting. The complexity of rank extraction mainly lies in the need for circulating information among the components of a system, whose interconnections might be difficult to implement by VLSI circuits [29]. A study of the structural aspects involving sorting time and the number of components is presented in [37]. Neural-network approaches use SOM's [38] or Hopfield networks [39]. VLSI circuits tackle the communication-exchange problem by limiting connections with pairwise comparisons; they then set up a hierarchy of progressive selections [10], [40]. Thanks to a better matching, local comparisons enhance robustness by higher accuracy [30]; on the other hand, the system area complexity tends to increase. In fact, the simplest architecture for HW sorting requires only one wire connecting several elementary cells. This approach is followed in [41], [42], where a reduction in corner and offset errors is also taken into account. For such structural features, a similar single-wire connection approach is adopted by the DSD of the VQ-training architecture.

The novel sorting circuitry described in the following best fits the overall external architecture and minimizes interconnections. The schema is based on Lazzaro's well-known WTA structure; the modular approach makes it possible to include different schemata from the literature that, for instance, might turn out to improve accuracy or speed (as presented in [41]). The DSD circuit operates iteratively: the length of the list K can be preset and sorting completes in $O(K)$ time. Each iteration includes two steps: 1) the current largest input value is detected by WTA competition; such a value is linearly represented at the output voltage, V_{out} and 2) the "winner" is removed from the list of competitors. For each output value, the circuit yields an analog representation of the value itself, a digital encoding of its rank on the sorted list, and a digital indexing of its position in the original set.

Each input current is handled by an associate elementary cell [Fig. 5(a)]; each cell integrates analog circuitry and a D-type positive-edge triggered flip-flop. A single wire V_{out} connects

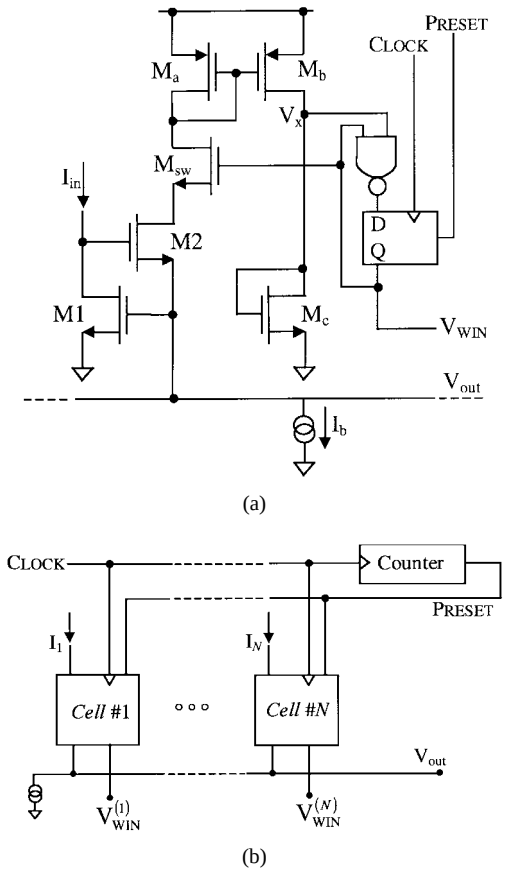


Fig. 5. The architecture of the sorting circuitry. (a) The elementary cell. (b) Cell-interconnection structure.

the gates of transistors $M1$ of all the cells [Fig. 5(b)] and is biased by a constant current I_b. At startup, a "low" PRESET pulse forces status Q to be high in all the cells.

Step 1: The identification of the largest input value at each iteration is supported by Lazzaro's WTA circuit [13], which has been chosen from several alternatives [43] mainly for its simplicity. The sets of transistor pairs $(M1, M2)$ of all the cells constitutes the basic WTA subcircuit. Thanks to the negative feedback with global competition on line V_{out}, the entire current I_b flows in transistor $M2$ of the cell with the largest input current. If the input currents in transistors $M1$ are suitably biased [44], V_{out} also yields an analog, linear, representation of the winning value.

Step 2: The sorting circuit exploits the consequent nonuniform distribution of bias current I_b. As long as switch M_{sw} is kept "on" by a high signal Q (default condition), the matched pair (M_a, M_b) and the load transistor M_c drive voltage V_x. If the cell is not the winner, no current is mirrored into M_c and V_x is low; V_x becomes high when the cell wins a competition. The logic circuitry exploits this information to remove the winner from the competition. At the next rising edge of CLOCK, the flip-flop status Q in the winning cell becomes low, and informs external circuitry about the current winner. Feeding back Q(low) to the flip-flop input inhibits further changes in the flip-flop status; this prevents the cell from re-entering competitions in the next iterations. At the same time, M_{sw} is turned off and virtually disconnects the WTA subcircuit $(M1, M2)$ from line V_{out}. As a result, the cell is removed from further competitions, and this condition holds until a PRESET pulse drives M_{sw} back on.

To sum up, the highest input value is mapped at V_{out} at the first iteration; as soon as the winning cell exits the competition, the second highest value is enabled to show up, and so forth; iterations are timed by clock strikes. As a result, the sorted list is represented by the sequence of potentials V_{out} at consecutive clock cycles. Programmable sorting $(k = 1, \cdots, K)$ is accomplished by presetting the counter to issue a PRESET pulse after K clock cycles.

D. Architectural Reasons for HW Implementation and Partial Sorting

The following analysis shows that the described architecture improves the efficiency of the training process substantially. To this end, we compare the timings of the analog architecture with the corresponding performance of an (ideal) architecture using serial digital circuitry. The analysis considers the phases significant to comparison, i.e., DCD and sorting.

In a standard implementation of the NG algorithm, the time taken by one iteration of the algorithm results from the sum of two terms

$$T_d = T_d^{(\text{dist})} + T_d^{(\text{sort})} > N \cdot d \cdot T_d^{(\text{sq})} + T_d^{(\text{cfr})} \cdot N \log_2 N \tag{13}$$

where $T_d^{(\text{sq})}$ and $T_d^{(\text{cfr})}$ indicate the times required to compute one term of summation (1) and to make a pairwise distance comparison, respectively. In order to evaluate $T_d^{(\text{sq})}$, we shall make an optimality assumption, involving a DSP-like architecture, running at clock period $T_d^{(\text{CK})}$ and able to best exploit the sequence of computation. Each term in summation (1) requires a difference, a product, and a sum; if the HW can execute a multiplication and a sum in a single clock strike, we obtain $T_d^{(\text{sq})} = 2T_d^{(\text{CK})}$. Moreover, we shall assume that the serial architecture can execute a comparison of two distances in one clock cycle, hence $T_d^{(\text{cfr})} = T_d^{(\text{CK})}$. Expression (13) involves a simplified model as, for example, it does not take into account the time required to swap two elements on the sorted list.

The performance attained by the analog architecture with partial sorting is expressed as

$$T_a = T_a^{(\text{ds})} + T_a^{(s)} = T_a^{(\text{sq})} + k \cdot T_a^{(\text{cfr})} \tag{14}$$

where the terms $T_a^{(\text{sq})}$ and $T_a^{(\text{cfr})}$ have the same meanings as above. The first term in expression (14) takes advantage of the on-chip parallelism as all sample-codevector distances are computed simultaneously, hence $T_a^{(\text{sq})} = T_a^{(\text{CK})}$; likewise, the sorting circuitry can yield a sorted quantity per clock cycle, hence $T_a^{(\text{cfr})} = T_a^{(\text{CK})}$. Experiments indicate a good setting for the partial sorted list: $k = 5$.

Standard values for low-rate VQ image coding are $d = 16$ and $N = 256$. The maximum operating speed of the analog circuitry is 2 MHz, hence $T_a^{(\text{CK})} = 500$ ns. If the digital implementation supports a DSP bus running at 100 MHz, $T^{(\text{CK})} = 10$ ns. Substituting these values into (13) and (14)

allows one to evaluate a lower bound to the architectural speed-up

$$\sigma = \frac{T_d}{T_a} > \frac{N(2d + \log_2 N)T_d^{(\mathrm{CK})}}{(k+1)T_a^{(\mathrm{CK})}} = 34. \qquad (15)$$

It is worth stressing that the above value allows quite optimistic assumptions about the serial implementation (for example, the whole architecture should run at the processor speed, no codevector swapping is accounted for in the sorting process, etc.). Therefore, the estimated bound (15) is substantially pessimistic, as measures obtained by various experiments with commercial DSP's yielded empirical values of the speed-up (15) always greater than 100. Anyway, the result obtained must be considered satisfactory, especially in view of the lower clock speed and of the lower complexitty of the overall architecture.

IV. Experimental Results

This section provides experimental support for both the theoretical derivations and the circuital implementations of the VQ-training algorithm. First, the method is compared with the NG algorithm, under the assumption that the latter behaves ideally; the overall distortion on the data set measures each model's effectiveness. Then, VLSI circuits are considered and evaluated.

A. Experimental Support for Partial Sorting

This section considers the quantization performance of the training algorithm. The goal of the experiments is to evaluate the specific effect of partial sorting to the overall VQ performance. Therefore, the algorithm results are compared with those obtained by the NG model, which requires a complete and exact sorting of codevectors. The comparison refers to both synthetic and real-world testbeds.

In the experiments, the length K of the partially sorted list was progressively increased and ranged from $K = 1$ to $K = 15$; in Section II, theory predicted that using larger values of K would be useless. The distortions conveyed by the resulting codebooks were related to the average distortion of NG, acting as the "reference" model. In order to remove statistical bias, thirty independent training runs were performed for each value of K and for the original NG algorithm $(K = N)$. Thus, the NG reference performance is represented by a range of values rather than by a single datum; likewise, results from the the partial-sorting method are summarized by their average, min, and max values. The comparison in performance between the proposed method and the ideal NG model can be expressed quantitavely by a percentage deviation δ, defined as

$$\delta = 100\frac{\hat{E}_K - \hat{E}_{\mathrm{NG}}}{\hat{E}_{\mathrm{NG}}} \qquad (16)$$

where $\hat{E}_K$ and $\hat{E}_{\mathrm{NG}}$ denote the average distorion of the partial-sort algorithm (with list length $= K$) and of the NG model, respectively.

The synthetic data distribution proposed by Fritzke makes it possible to visually evaluate codevector positions [6], [7], and allows comparisons with previous results in the literature. The testbed included 4483 two-dimensional samples, whose spatial distribution is shown in Fig. 6(a); a plastic algorithm [17] evaluated the appropriate number of codevectors $(N = 30)$. Fig. 6(b) displays averaged results in terms of percent deviation from the reference distribution. The graph clearly shows that, if K increases, distortion converges to that obtained by NG; at the same time, the distribution of results shrinks. This confirms theoretical expectations, i.e., partial sorting approximates ideal NG fairly well, even for small values of K; moreover, the confidence in such approximation increases accordingly.

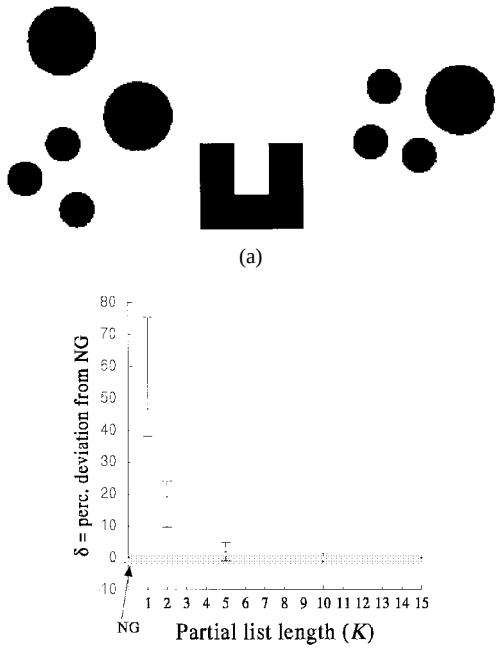
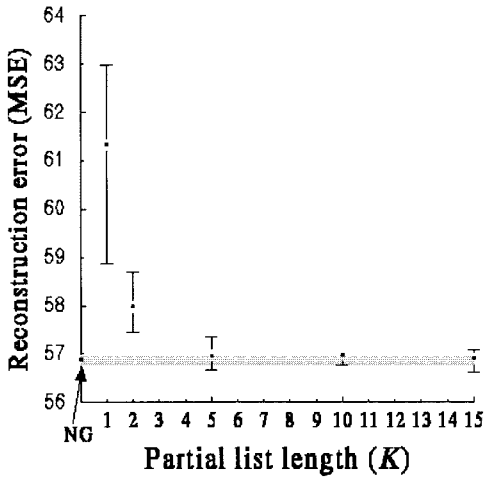


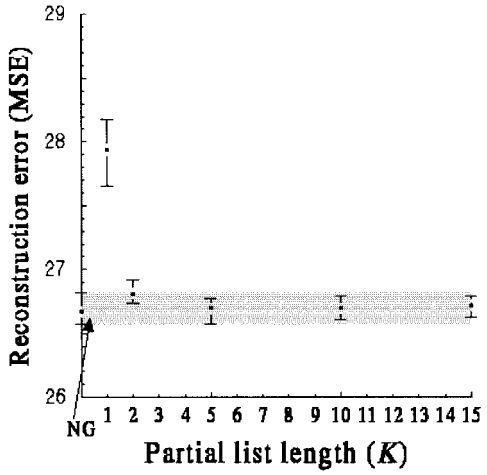
Fig. 6. Artificial-testbed results for partial sorting. (a) Fritzke's synthetic testbed for evaluation of clustering ability [6]. (b) Deviation from NG performance; the grey area marks the distribution of ideal NG. Small values of K (e.g., $K = 5$) provide good approximations for the ideal performance.

The validation of the VQ-training algorithm eventually tackled a real testbed, i.e., image coding. This application is technically significant as VQ methods can attain remarkable compression ratios and yet maintain a satisfactory picture quality, especially at very low bit rates [18]. In a VQ-based image-coding system, a picture is split into elementary sub-blocks, playing as samples in the quantized space. Typical block sizes cover 4×4 or 8×8 pixel grids. In the experiments, standard, gray level (8 bpp) images of size 512×512 were used, and blocks of both sizes (4×4 and 8×8) were tested. In all cases, codebooks included 256 codevectors, and this setting was worked out again by specific plastic-network models [17].

The present analysis only focuses on a distortion comparison with the NG model; hence, we refer the reader to [17] for both a qualitative assessment of the image-coding methodology and an extensive treatment of generalization performance. The latter issue is a crucial problem in VQ algorithm testing: after completing a training phase and creating a codebook,



(a)



(b)

Fig. 7. Image-coding results for partial sorting. The greyed area marks the distribution of ideal NG. (a) MSE results for 8 × 8 pixel blocks. (b) MSE results for 4 × 4 pixel blocks.

the distortion performance must be evaluated on codevectors outside the training set. This is especially true in image coding, and represents a true test of a robust codebook. In fact, the codeword positions determined by the partial-sorting method and by the NG are very close; hence, the two approaches exhibited equivalent generalization performance.

Fig. 7(a) and (b) display results for the block sizes 4 × 4 and 8 × 8, respectively. Due to the applicative nature of the image-coding testbed, the direct metric of performance is reported on the y axis. Therefore, the graphs compare reconstruction MSE for the experiment; this indicates the tradeoff in performance for different values of K. The reference (NG) distribution is wider for the 4 × 4 blocks than for the 8 × 8 ones, as the 4× 4 block space is sampled much more intensely. The very narrow gray band in Fig. 7(a) indicates that NG results deviated from their average value by at most 0.15%; in other words, the NG algorithm virtually found one minimum point of the distortion cost. Partial-sorting results slightly exceeded the NG distribution boundaries never more than 0.1% for $K > 10$; negative deviations suggest that partial sorting is less sensitive to the annealing process. The average, min, and max MSE values for the 8 × 8 block space, however, always lay within the NG range for $K > 20$. To summarize, in the image-coding experiments, empirical evidence again confirms theoretical predictions: the distributions of distortion always converge to those of ideal NG when K increases, whereas the associate confidence intervals shrink progressively.

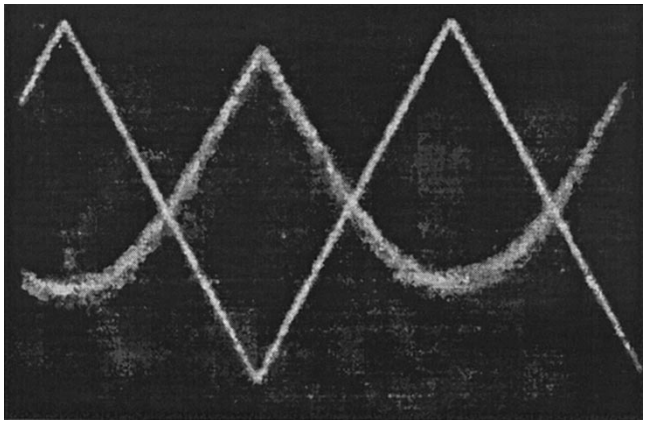


Fig. 8. Output of the square-of-difference circuit for a triangular input signal.

It is worth noting that, in all testbeds, the upper bound to K was always set *a priori* to 15, yielding satisfactory results (i.e., fairly good approximations for the NG performance) anyway. In other words, the length of the partially sorted list appears invariant to the specific quantization problem. This property seems the best experimental confirmation of the importance analysis made in Section II.

B. DCD Results

Experimental verifications on the DCD chip have been performed to assess proper implementation of the distance-computing circuitry and overall performance.

The distance-computing block, a square-of-difference circuit, has been separately tested and compared with the required behavior. To this purpose, a stand-alone block has been added to the chip. The squaring function reveals good symmetry and little overall error versus the theoretical characteristic. We can observe that a symmetric behavior (polynomial characteristic with even degree) is more important than exact quadratic behavior in this specific application, namely, DCD. The output current ranges from 3.39 to 3.52 mA for an input voltage ranging from 2 to 2.95 V (slightly less than that of the theoretical model). The best matching quadratic characteristic is approximated with an average square error of about 5%, and maximum deviation of 0.01 mA. Fig. 8 illustrates the response of the circuit to a triangular waveform test signal.

The overall performance of the chip has been assessed, with the aid of a dedicated test board, to enable evaluation of the single-chip properties. Experiments have been performed with 512 × 512-pixel images and 40-vector codebooks. The throughput of the DCD chip is constrained mainly by the on-chip decision circuitry (WTA), so that, when used as distance-computation block only, its speed performances can be fully exploited. Experimental verifications result in an estimated throughput of about 10^6 input vectors/s. We remark

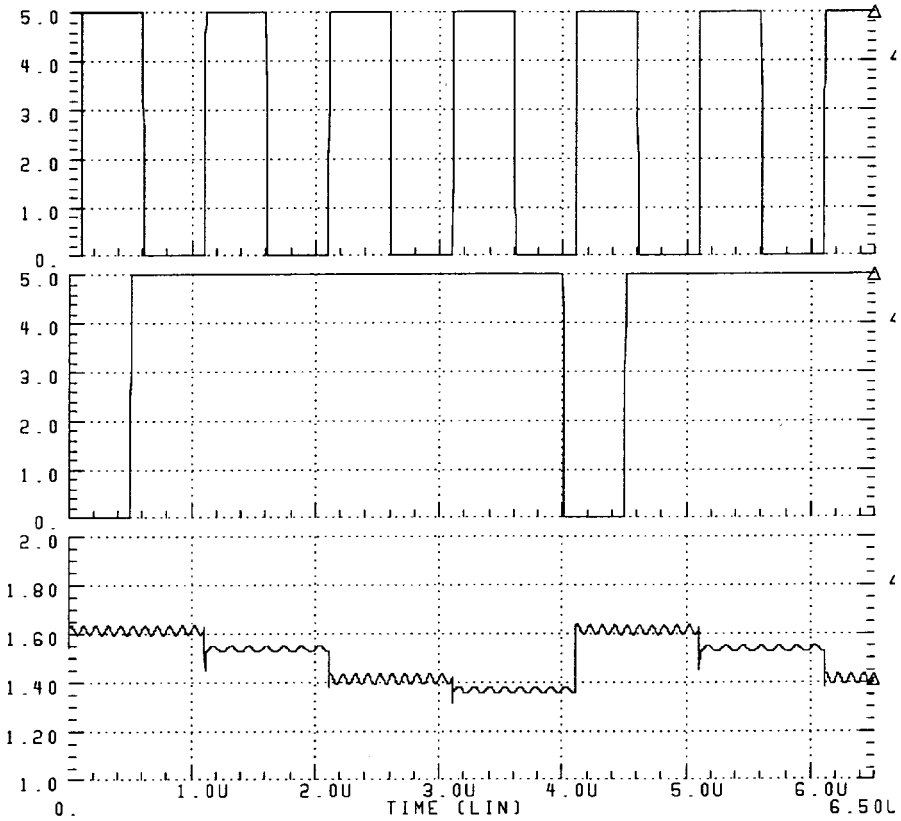


Fig. 9. Simulation results of a sorting circuit with four input lines: Clock CK (top); PRESET (middle); $V_{\rm out}$ (bottom).

that internal operation is fully parallel, so that the required clock signal can work at relatively low frequencies, since a whole pattern is processed in a single clock cycle.

C. Experimental Evaluation of DSD

The DSD was simulated at the layout level by using HSPICE lev.13 with 0.8-μm technology by Austria Mikro Systeme (AMS); power supply was 5 V. The circuit was tested under stressing conditions (up to 64 input lines) and always proved effective and accurate. The aspect ratios for the analogue subcircuit were $(W/L)_1 = (W/L)_2 = 25/5$; $(W/L)_{\rm sw} = 5/2$; $(W/L)_a = (W/L)_b = 30/2$; $(W/L)_c = 1/2$. The digital subcircuitry used standard CMOS configurations. Input currents were in the range $[50\,140]$ μA to ensure compatibility with the outputs of the DCD circuitry; the bias current was $I_b = 30$ μA; the chip clock frequency was 2 MHz. An appealing feature of the presented circuit is a relative insensitivity to design parameters, thanks to the integration of analog and digital subcircuits. Conversely, power consumption represents a drawback; this mainly relates to the current-mode WTA mechanism, especially when its above-threshold biasing [27], [44], is considered. In the present implementation, power requirement varies in the range 1–3 mW/cell.

For simplicity of notation and interpretation, Fig. 9 presents a demonstration involving four cells with a clock at 1 MHz; the associate input currents (in μA) are $I_{\rm in}^{(1)} = 80+3\cdot\sin(2\pi\cdot 8\cdot 10^6)$; $I_{\rm in}^{(2)} = 120+3\cdot\sin(2\pi\cdot 7\cdot 10^6)$; $I_{\rm in}^{(3)} = 90+5\cdot\sin(2\pi\cdot 10\cdot 10^6)$; $I_{\rm in}^{(4)} = 140+5\cdot\sin(2\pi\cdot 10\cdot 10^6)$. The input exhibits a high-frequency component that has been introduced only for demonstration purposes for the reader's convenience. It allows the reader to have an immediate visual perception of the system-correct sorting; the sequence of analog outputs at $V_{\rm out}$ witnesses the correct functioning of the circuit and the linear mapping of input/output values.

The device is currently being fabricated with the same technology as used for the DCD device, and the apparent limitation on the number of elementary cells can be easily overcome by including multiple devices in a common architecture [45]. However, other schemata for the WTA subnetwork are being examined, especially in order to improve accuracy by multistage WTA subcircuits.

V. Discussion and Conclusion

The technical interest in VQ models stems from their remarkable performances in difficult applications with high-dimensional data representations and oversampled data spaces. Image coding at very low bit rates is a prototype for such applications. At the same time, these very domains make VQ quite demanding in terms of computational costs of training and real-time performance. If this fact poses a basis for supporting VQ training in HW, not all theoretical models are equally prone to a circuit implementation. The amount of involved computation and the interconnection structure among codewords (or neurons, in neural-network terminology) sometimes hinder critically a direct porting of algorithms to circuits.

Architectural parallelism and distributed management of information provide a viable approach to overcoming those

issues. In HW, the former calls for several, independent units entrusted with specific computational ability; the latter confines data to the local level and minimizes the circulation of information by reducing the number of interconnecting wires.

The research presented in this paper accounts for such aspects with a VQ-training algorithm specifically aimed at HW implementation. Specific features of the method are codeword-level local computation and minimal circulation of information by partial sorting. The theoretical analysis made in Section II primarily serves to prove the method consistency in terms of quantization quality, as compared with a similar algorithm (NG) that is known to provide a notable improvement in performance.

Thus, the overall approach matches the above guidelines for effective porting: using analog technology for the distance-computation device optimizes silicon area by reducing the occupation of single cells. Likewise, an elementary schema implements partial sorting and minimizes interconnection circuitry down to one wire. The two devices operate in conjunction with external digital circuitry for weight storage/adjustment to permit flexible training strategies. The integrated approach makes it possible to best exploit analog components to reduce bandwidth bottlenecks and, at the same time, to benefit from digital circuitry in permanent storage and from programming facility.

From a qualitative perspective, the theoretical analysis and the related experimental results appear complementary. The algorithm's effectiveness in optimizing quantization noise has been attested substantially by showing the method equivalence to a "reference" model described in the literature. More importantly, a crucial theoretical result lies in proving that partial sorting operates independently of a specific application, hence no on-field tuning is required.

These results justify the HW-implementation effort, which has led first to the design and realization of the DCD. This device supports the core of the overall quantization process as its internal parallel architecture boosts the timing performance by reducing the bandwidth required for computation. The testing process of the first production run of the VLSI chip has confirmed the robustness and accuracy of the original design, and has opened stimulating prospects for the next engineering step. This redesign phase will mainly aim to upgrade production by using a better technology (AMS 0.7μm) in order both to incorporate a larger number of codevectors and to increase operating speed.

ACKNOWLEDGMENT

The authors wish to thank the reviewers for carefully reading the paper; their precise and constructive remarks greatly contributed to improving the overall quality of the work.

REFERENCES

[1] *IEEE Trans Inform Theory*, Special Issue on Vector Quantization, vol. IT-28, Mar. 1982.
[2] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. COM-28, pp. 84–95, Jan. 1980.
[3] Y. Q. Chen, R. I. Damper, and M. S. Nixon, "On neural-network implementations of k-nearest neighbor pattern classifiers," *IEEE Trans. Circuits Syst. I*, vol. 44, pp. 622–629, 1997.
[4] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'Neural Gas' network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Neworks*, vol. 4, pp. 558–569, 1993.
[5] T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed. New York: Springer-Verlag, 1989.
[6] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Inform. Processing Syst. 7 NIPS-7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: MIT Press, 1995.
[7] B. Fritzke, "Growing grid—A self-organizing net-work with constant neighborhood range and adaptation strength," *Neural Processing Lett.*, vol. 2, no. 5, pp. 1–5, 1995.
[8] D. DeSieno, "Adding a conscience to competitive learning," in *Proc. IEEE Int. Conf. Neural Networks*, San Diego, CA, 1988, vol. I, pp. 117–124.
[9] M. M. Van Hulle, "Kernel-based equiprobabilistic topographic map formation," *Neural Computation*, vol. 10, pp. 1847–1871, 1998.
[10] Y. Fang, M. A. Cohen, and T. G. Kincaid, "Dynamics of a winner-take-all neural network," *Neural Networks*, vol. 9, pp. 1141–1154, July 1996.
[11] Y. H. Tseng and J. L. Wu, "On a constant-time, low-complexity winner-take-all neural network," *IEEE Trans. Comput.*, vol. 44, pp. 601–603, 1995.
[12] T. C. Hsu and S. D. Wang, "k-winners-take-all neural net with $\Theta(1)$ time complexity," *IEEE Trans. Neural Networks*, vol. 8, pp. 1557–1561, 1997.
[13] J. Lazzaro, R. Ryckebush, M. A. Mahowald, and C. Mead, "Winner take all networks of $O(n)$ complexity," *NIPS 2*. Denver, CO: Morgan Kaufmann, 1989, pp. 703–711.
[14] J. A. Starzyk and X. Fang, "CMOS current mode winner-take-all circuit with both excitatory and inhibitory feedback," *Electron. Lett.*, vol. 29, no. 10, pp. 908–910, 1993.
[15] S. Smedley, J. Taylor, and M. Wilby, "A scalable high-speed current-mode winner-take-all network for VLSI neural applications," *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 289–291, 1995.
[16] T. Serrano and B. Linares-Barranco, "A modular current-mode high-precision winner-take-all circuit," *IEEE Trans. Circuits Syst. II*, vol. 42, pp. 132–134, 1995.
[17] S. Ridella, S. Rovetta, and R. Zunino, "Plastic algorithm for adaptive vector quantization," *Neural Computing and Applicat.*, vol. 7, no. 1, pp. 37–51, 1998.
[18] N. Nasrabadi and R. King, "Image coding using vector quantization: A Review," *IEEE Trans. Commun.*, vol. 11, pp. 957–971, Aug. 1988.
[19] Y. He and U. Çilingiroglu, "A charge based on-chip adaptation Kohonen neural network," *IEEE Trans. Neural Networks*, vol. 4, pp. 462–468, Mar. 1993.
[20] D. Macq, M. Verleysen, P. Jespers, and J. D. Legat, "Analog implementation of a Kohonen map with on-chip learning," *IEEE Trans. Neural Networks*, vol. 4, pp. 456–461, 1993.
[21] K. Dezhgosha, M. M. Jamali, and S. C. Kwatra, "A VLSI architecture for real-ime image coding using a vector quantization based algorithm," *IEEE Trans. Signal Processing*, vol. 40, pp. 181–189, 1992.
[22] R. Jain, A. Madisetti, and R. L. Baker, "An integrated circuit design for pruned tree-search vector quantization encoding with an off-chip controller," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, pp. 147–158, 1992.
[23] T. Serrano-Gotarredona and B. Linares-Barranco, "A real-time clustering microchip neural engine," *IEEE Trans. VLSI Syst.*, vol. 4, pp. 195–209, 1996.
[24] A. Lipman and W. Yang, "VLSI hardware for example-based learning," *IEEE Trans. VLSI Syst.*, vol. 5, pp. 320–328, 1997.
[25] J. Lubkin and G. Cauwenberghs, "A micropower learning vector quantizer for parallel analog-to-digital data compression," presented at IEEE ISCAS'98, Monterey, CA, 1998.
[26] F. Ancona, S. Rovetta, and R. Zunino, "Efficient technique for implementing an image-compression neural algorithm on concurrent-multiprocessor architectures," *Int. J. Eng. Applicat. Artificial Intell.*, vol. 10, no. 6, pp. 573–580, 1998.
[27] F. Ancona, G. Oddone, S. Rovetta, G. Uneddu, and R. Zunino, "A vector quantization circuit for trainable neural networks," in *Proc 3rd Int. Conf. Electron. Circuits and Systems ICECS'96—Rodos*, vol. 2, pp. 1131–1134.
[28] G. Oddone, S. Rovetta, S. Uneddu, and R. Zunino, "Mixed analog-digital circuit for linear-time programmable sorting," in *Proc. IEEE ISCAS'97*, Hong Kong, vol. III, pp. 1968–1971.
[29] S. S. Lin and S. H. Hsu, "A low-cost neural sorting network with O(1)

time complexity," *Neurocomputing*, vol. 14, pp. 289–299, 1997.
[30] A. Demosthenous, S. Smedley, and J. Taylor, "A CMOS analog winner-take-all network for large-scale applications," *IEEE Trans. Circuits Syst. I*, vol. 45, pp. 300–304, 1998.
[31] G. Cauwenberghs and V. Pedroni, "A low-power CMOS analog vector quantizer," *IEEE J. Solid-State Circuits*, vol. 32, pp. 1278–1283, 1997.
[32] W. C. Fang, C. Y. Chang, B. J. Sheu, O. T. C. Chen, and J. C. Curlander, "VLSI systolic binary tree-searched vector quantizer for image compression," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 33–44, Jan. 1994.
[33] K. Tsang and B. W. Y. Wei, "A VLSI architecture for a real-time code book generator and encoder of a vector quantizer," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 360–364, 1994.
[34] W. C. Fang, B. J. Sheu, O. T. C. Chen, and J. Choi, "A VLSI neural processor for image data compression using self-organization networks," *IEEE Trans. Neural Networks*, vol. 3, pp. 506–518, 1992.
[35] J. S. Pan, F. R. McInnes, and M. A. Jack, "Fast clustering algorithms for vector quantization," *Pattern Recognition*, vol. 29, no. 3, pp. 511–518, 1996.
[36] S. Satyanarayana and K. Suyama, "Resistive interpolation biasing: A technique for compensating linear variation in array of MOS current sources," *IEEE J. Solid-State Circuits*, vol. 30, pp. 595–598, 1995.
[37] G. V. Russo and M. Russo, "A novel class of sorting networks," *IEEE Trans. Circuits Syst. I*, vol. 43, pp. 544–552, 1996.
[38] M. Budinich, "Sorting with self-organizing maps," *Neural Computation*, vol. 7, pp. 1188–1190, 1995.
[39] J. Wang, "Analysis and design of an analog sorting network," *IEEE Trans. Neural Networks*, vol. 6, pp. 962–971, 1995.
[40] K. Urahama and T. Nagao, "Direct analog rank filtering," *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 385–388, 1995.
[41] I. E. Opris, "Analog rank extractors," *IEEE Trans. Circuits Syst. I*, vol. 44, pp. 1114–1121, 1997.
[42] I. E. Opris, "Rail-to-rail multiple-input min/max circuit," *IEEE Trans. Circuits Syst. II*, vol. 45, pp. 137–140, 1998
[43] E. Sanchez-Sinencio and Z. S. Gunay, "CMOS winner-take-all circuits: A detail comparison," in *Proc. ISCAS'97*, Hong Kong, vol. I, pp. 41–44.
[44] Y. He and E. Sanchez-Sinencio, "Min-net winner-take-all CMOS implementation," *Electron. Lett.*, vol. 29, no. 14, pp. 1237–1239, 1993.
[45] F. Ancona, G. Oddone, S. Rovetta, S. Uneddu, and R. Zunino, "VLSI architectures for programmable sorting of analog quantities with multiple-chip support," in *Proc. 7th IEEE Great Lakes Symp. on VLSI*, Urbana, IL, Mar. 1997, pp. 94–99.

Stefano Rovetta (M'98) received the Laurea degree in electronic engineering in 1993, and the Ph.D. degree in models, methods, and tools for electronic and electromagnetic systems in 1997, both from the University of Genoa, Italy.

He is currently a Postdoctoral Researcher with the Electronic Systems and Networking Group, Department of Biophysical and Electronic Engineering, University of Genoa, where he also teaches Electronics. He is an Invited Professor of Operating Systems at the University of Sienna, Italy. His research interests include electronic circuits and systems, and neural network theory, implementation, and applications.

Rodolfo Zunino (S'90–M'90) recieved the Laurea degree in electronic engineering from Genoa University, Italy, in 1985.

From 1986 to 1995, he was a Research Consultant with the Department of Biophysical and Electronic Engineering, Genoa University, where he is now an Assistant Professor of Industrial Electronics. His main scientific interests include electronic systems for neural networks, distributed control, and methods for data representation and processing.