An Analog Neural Network Implementation in Fixed Time of Adjustable-Order Statistic Filters and Applications

Mohammed Mestari

***Abstract*—In this paper, we show a neural network implementation in fixed time of adjustable order statistic filters, including sorting, and adaptive-order statistic filters. All these networks accept an array of N numbers $X_i = S_{Xi}M_{Xi}2^{E_{Xi}}$ as input (where S_{Xi} is the sign of X_i, M_{Xi} is the mantissa normalized to m digits, and Ex is the exponent) and employ two kinds of neurons, the linear and the threshold-logic neurons, with only integer weights (most of the weights being just $+1$ or -1) and integer threshold. Therefore, this will greatly facilitate the actual hardware implementation of the proposed neural networks using currently available very large scale integration technology. An application of using minimum filter in implementing a special neural network model neural network classifier (NNC) is given. With a classification problem of l classes $C_1, C_2, \ldots, C_1$, NNC classifies in fixed time an unknown vector to one class using a minimum-distance classification technique.**

***Index Terms*—Adjustable-order statistic filters (AOSFs), minimum-distance classification (MDC) technique, neural networks, neural network classifier (NNC), sorting filter.**

I. Introduction

ORDER statistic filtering is a technique extensively used in pattern recognition and image processing (PRIP) applications [4], [9], [11], [12], [38], [49]–[52], [55], [58], [65], [68], [71], [79]. During the past decades, considerable efforts have been devoted to developing special computer architectures for PRIP applications [7], [41], [42], [54], [63], [66], [69]. Recent advances in very large scale integration (VLSI) microelectronic technology have triggered the idea of implementing PRIP algorithms directly in specialized hardware chips. Many attempts have been made to develop special VLSI devices for such purposes. It is of certain importance and interest to develop a hardware model of high processing speed that can be used as a building block for implementing any order statistic filter (OSF), including sorting and adaptive OSFs (called comparison and selection filters [38]). The main task of the order statistic filtering is to find the kth-order statistic of an input array, defined as being the kth largest element in the array. This technique finds application in telecommunications, particularly for controlling data packet switches [8], [10]–[13]. In [34], a member of the OSF family shows applications in VLSI auditory and visual systems, while in [20], another filter of OSF family as an analog decoder of error-correcting codes is proposed. OSFs are mostly implemented in software [4], [6], [19], [24], [28], [30], [43], [56], [59], [63]. [71], [76], [79], whereas hardware implementations are designed only for specific members, particularly the median and maximum filters, of the OSF family [18], [29], [49], [50], [52], [62], [67], [86].

In this paper, we propose a neural network model, the adjustable order statistic filter (AOSF), which is to be used as a building block for implementation of any member of OSF family. The function of the AOSF is to find in fixed time the kth largest element of an array of N numbers $X_i = S_{X_i}M_{X_i}2^{E_{X_i}}$, where S_{X_i} is the sign of X_i and is coded on 1 bit ($S_{X_i} = 0$ if X_i is positive or zero, and $S_{X_i} = 1$ otherwise); M_{X_i} is the mantissa normalized to m digits and is coded on m bits; and E_{X_i} is the exponent and is coded on p bits; and i, k are integers, so that $1 \leq i \leq N$ and $1 \leq k \leq N$.

The approach behind constructing AOSF differs from the conventional approach used in the field of neural networks [33], [81]–[85]. To solve a specific problem, instead of taking a general-purpose network and applying it by learning, we tailor-make a dedicated network. Our primary concern is how to organize neurons into a network so that it can solve a specific problem, with an emphasis on fully utilizing the massive parallelism property offered by neural networks. This approach is useful for solving problems where exact analytic solutions are known or derivable. Two types of neurons are employed in AOSF (linear and threshold-logic neurons). Both types have already been implemented in the past using analogue electronic circuits. They both assume the well-known linear sum neuron model and differ only in their activation functions. A typical implementation of the linear neuron uses a linear operational amplifier [25]. The linear neuron may also be implemented using summing amplifiers (adders) [17] by restricting the value range of the inputs to be within the linear range of the amplifier. As for the threshold-logic neuron, many different schemes of implementation have been reported. All such neurons have been used in constructing various kinds of neural networks [3], [17], [21], [26], [32], [44], [49]–[53].

An important application of using AOSF $(k = N)$ for implementing a special neural network model neural network classifier (NNC) is described. The NNC performs classification using minimum-distance technique. Minimum-distance classification (MDC) is a simple yet powerful technique widely used in statistical recognition, clustering, and other applications.

All neural networks considered in this paper have a feed-forward structure with two kinds of neurons, linear and threshold-

Manuscript received May 22, 2002; revised September 3, 2002.
The author is with ENSET Mohammedia, Mohammedia, 20800 Morocco.
Digital Object Identifier 10.1109/TNN.2003.820656

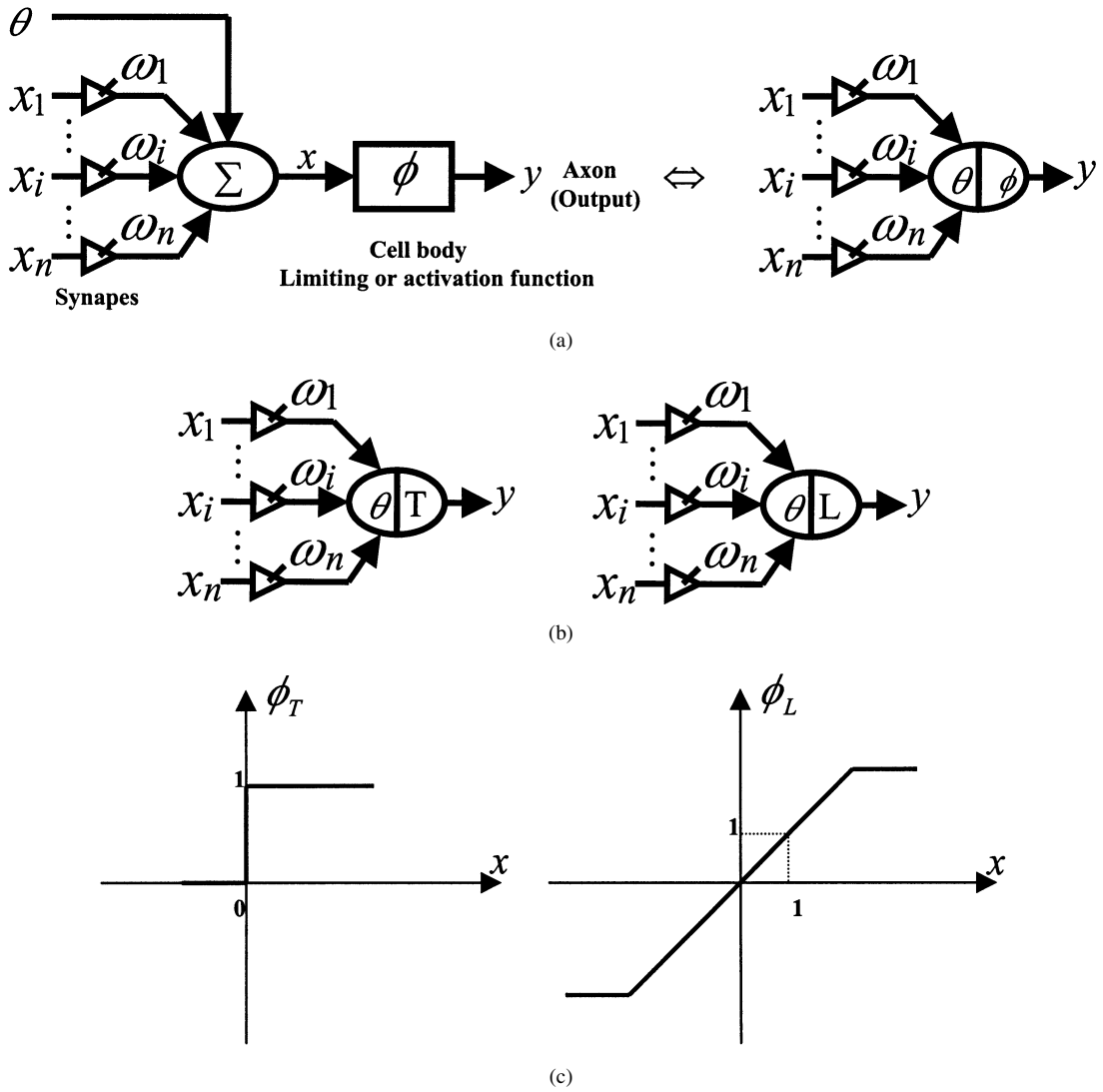


Fig. 1. (a) Simplified functional model of an artificial basic neuron cell. (b) Schematic representations of the threshold-logic neuron and the linear neuron, respectively. (c) Two possible unipolar transfer characteristics.

logic neurons. These networks have a very simple configuration, the connection strengths between the neurons are all fixed, most of them being just $+1$ or -1, which makes hardware implementation easy and straightforward. The modularity and the regularity of the networks' architecture make them suitable for VLSI implementation.

The processing time of each network herein proposed is constant. As the size of the input array increases, only the number of neurons in each layer increases, not the number of layers themselves. Therefore, the network's total processing time remains constant, irrespective of the size of the input array. This is in contrast with conventional digital hardware implementation, where the processing time increases along with the input size. Although constant processing time is achievable using unlimited fan-in logic gates, the circuit size grows exponentially as the size of the input increases [15], [23]. The circuit size of AOSF, however, only grows quadratically.

The rest of the paper is organized as follows. In Section II, we describe the construction of the basic neural network AOSF. Section III is devoted to the application of the AOSF in implementation of various order statistic filters, including sorting and adaptive order statistic filters. An important example of application of AOSF $(k = N)$ to implementing an NNC is given at the end of this section. Section IV contains the conclusion.

II. Construction of the AOSF

In this section, we develop the basic neural network, AOSF, which is essential for construction of all neural networks herein proposed, but first, we describe the neurons employed here.

A. Neurons Used

AOSF employs two kinds of neurons, both of which are commonly used in neural network applications [21], [32], [44], [49]–[53]. The only difference between the two is in their activation function: one employs the linear activation function and the other the threshold-logic activation function. Their schematic representations are shown in Fig. 1(b), where y is the output.

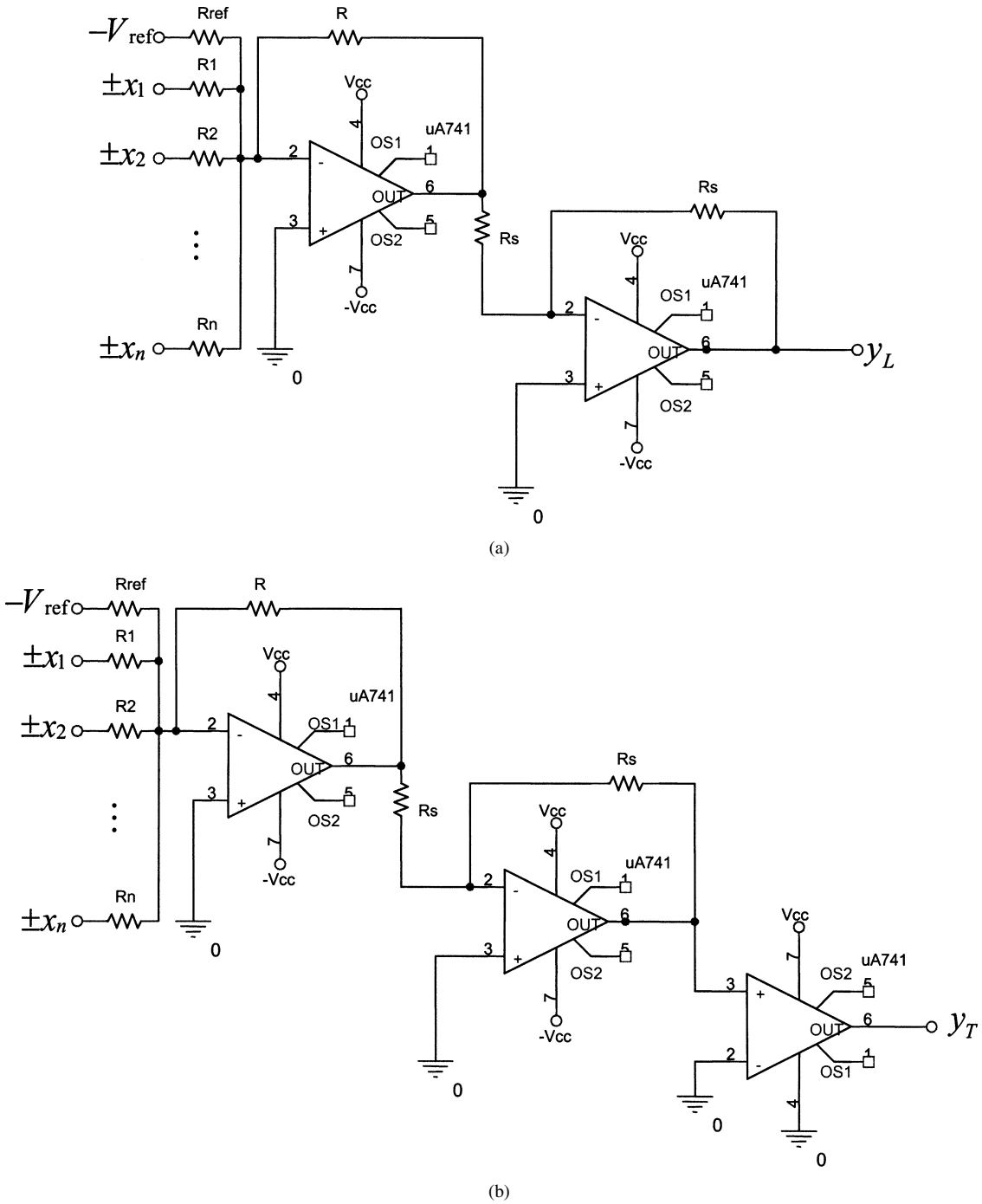


Fig. 2. Circuit implementation using resistors and operational amplifiers: (a) linear neuron and (b) threshold-logic neuron.

These two kinds of neurons sum the n weighted inputs and pass the result through a nonlinearity according to

$$y = \Phi\left(\sum_{i=1}^{n} \omega_i x_i - \theta\right) \tag{1}$$

where Φ is a limiting or nonlinear transfer characteristic, called an activation function; $\theta(\theta \in \mathbb{R})$ is the external threshold, also called an offset or bias; ω_i are the synaptic weights or strengths; x_i are the inputs $(i = 1, 2, \ldots, n)$, n is the number of inputs, and y represents the output [cf. Fig. 1(a)].

The threshold-logic neuron model [see Fig. 1(b)] uses only the binary (hardlimiting) function [see Fig. 1(c)]. In this model, a weighted sum of all inputs is compared with a threshold θ. If this sum exceeds the threshold, the neuron output is set to "high value" or to "low value" according to

$$\Phi_T(x) \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $x = \sum_{i=1}^{n} \omega_i x_i - \theta$, and Φ_T is the threshold-logic activation function or binary activation function [cf. Fig. 1(c)].

TABLE I
SERIES OF TESTS CARRIED OUT ON THE ELECTRONIC CIRCUITS IN FIG. 2 (a) AND (b)

Weighted sum of input signals, x	Linear neuron output voltage y_L	Threshold-logic neuron output voltage y_T
4.5468	4.54	4.61351
4.59868	4.60	4.61352
4.64548	4.64	4.61353
4.51857	4.52	4.61351
3.92399	3.93	4.61345
2.44654	2.46	4.61327
0.7389	0.752	4.61307
0.53596	0.547	4.61304
0.14688	0.157	4.61274
0.10087	0.111	4.61207
0.01013	0.0203	4.60012
-0.13846	-0.129	2.77385
-0.21054	-0.201	0.79898
-0.27743	-0.268	0.39852
-0.34042	-0.331	0.38817
-0.40973	-0.401	0.38727
-0.48818	-0.479	0.38701
-0.56451	-0.556	0.38709
-0.71071	-0.703	0.38711
-0.97613	-0.969	0.3871
-1.39248	-1.39	0.38709
-1.75729	-1.76	0.38709

In the case of the linear activation function [see Fig. 1(b)], the output y is given by

$$\Phi_L(x) \stackrel{\text{def}}{=} x \tag{3}$$

where Φ_L is the linear activation function [see Fig. 1(c)] defined by $\Phi_L(x) = x$ with $x = \sum_{i=1}^{n} \omega_i x_i - \theta$.

Both kinds of neurons, threshold and linear, have already been implemented in the past using analog electronic [17], [25].

Fig. 2(a) shows an example of electronic implementation of a linear neuron by employing resistors and operational amplifiers. The output voltage y_L can vary from $-V_{cc}$ to $+V_{cc}$. Applying Kirchhoff's current law (KCL), we obtain

$$y_L = \sum_{i=1}^{n} \left(\pm \frac{R}{R_i}\right) x_i - \frac{R}{R_{\text{ref}}} V_{\text{ref}} \tag{4}$$

with the voltage constraints

$$|y_L| \leq +V_{cc}. \tag{5}$$

Equation (4) can be written in the compact form

$$y_L = \sum_{i=1}^{n} \omega_i x_i - \theta \tag{6}$$

where $\omega_i = (\pm R/R_i)$ and $\theta = (R/R_{\text{ref}})V_{\text{ref}}$.

A model of the threshold-logic neuron can be built using traditional electronic circuit components as shown in Fig. 2(b). The comparator output voltage y_T replaces the output signal of real neuron. The threshold-logic activation function is naturally provided by the saturating characteristic of the amplifier used as comparator. By applying KCL, we obtain the expression

$$y_T = \begin{cases} +V_{cc}, & \text{if } \sum_{i=1}^{n} \left(\pm \frac{R}{R_i}\right) x_i - \left(\frac{R}{R_{\text{ref}}}\right) V_{\text{ref}} \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

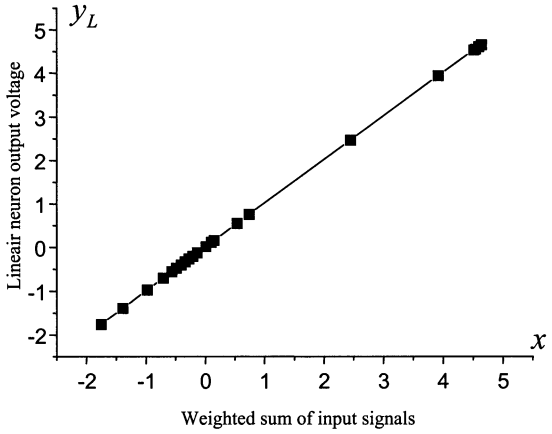
This can be written in the compact form

$$y_T = \begin{cases} +V_{cc}, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

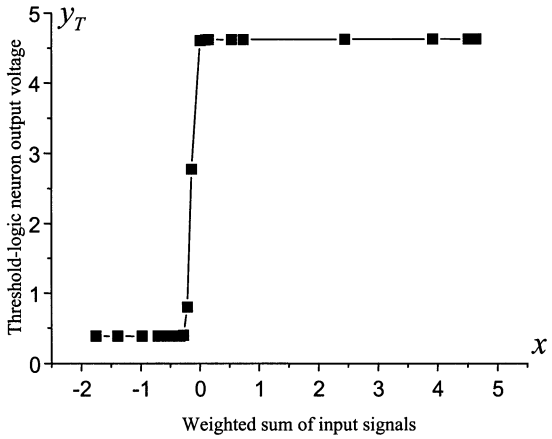
where $x = \sum_{i=1}^{n} \omega_i x_i - \theta$ with $\omega_i = \pm(R/R_i)$ and $\theta = (R/R_{\text{ref}})V_{\text{ref}}$.

A series of tests carried out on the electronic circuits in Fig. 2(a) and (b) (see Table I) has enabled tracing of linear and threshold-logic transfer characteristics [see Fig. 3(a) and (b)]. Passage from 0 to $+V_{cc}$ of the curve representing the output comparator voltage y_T according to the weighted sum of input signals $x_i (i = 1, 2, \ldots, n)$, $x = \sum_{i=1}^{n} \omega_i x_i - \theta$ is not instantaneous, as foreseen by (8) and the definition of a threshold-logic neuron; this is essentially due to the hysteresis (positive feedback) and the slew rate (13 $\text{V}/\mu\text{s}$) of the op-amplifier (μA741) used as comparator, and also to the gain-bandwidth of op-amplifiers placed before the comparator. We can remedy this problem by using a current feedback op-amplifier whose slew rate reaches (1000 $\text{V}/\mu\text{s}$), such as AD844 [73].

Current-mode signal processing offers several advantages when used in neural circuits. One of the most obvious advantages is that the summing of many signals is most readily accomplished when those signals are currents. Other advantages are increased dynamic range in future VLSI technologies, which are expected to see power supply reductions, high-speed signaling at low impedance nodes due to minimal capacitive



(a)



(b)

Fig. 3. Output curves of the linear and threshold-logic neurons, respectively: (a) linear transfer characteristic and (b) threshold-logic transfer characteristic.

charging/discharging, and extension of linear ranges in transistor circuits. The amount of linearity can be increased by representing signals as current differences in transistors and canceling common-mode nonlinear terms at virtual short inputs of operational transconductance amplifiers. Increased linearity is achieved using more complex cells.

In what follows, our primary concern is how to organize the linear and threshold-logic neurons into a network so that it can solve a specific problem, with an emphasis on full utilization of the massive parallelism property offered by neural networks.

B. Some Basic Functions

In this subsection, we introduce special functions that are essential for construction. First, however, we give the representation employed to represent the elements of input array X.

With a view to making the neural models proposed in this paper adaptable and to then facilitate their incorporation into digital calculators, we will employ the coding used in the majority of present-day computers to represent the elements of input array X.

Let X_i be an element of the input array $X(i = 1, 2, \ldots, N)$. Each element X_i of X is represented in a single way by the triplet (S_{X_i}, M_{X_i}, E_{X_i}), as follows:

$$X_i = S_{X_i} M_{X_i} 2^{E_{X_i}} \tag{9}$$

where

- S_{X_i} designating the sign of Xi is coded on 1 bit ($S_{X_i} = 0$ if X_i is positive or zero, and $S_{X_i} = 1$ otherwise);
- M_{X_i} designating the mantissa normalized to m digits is coded on m bits (M_{X_i} is a real number: $(1/2) \leq M_{X_i} < 1$);
- E_{X_i} designating the exponent is coded on p bits (E_{X_i} is a positive, negative, or zero integer).

The mantissa normalized to m digits M_{X_i} is represented in (binary) base 2 by

$$M_{X_i} = \sum_{j=1}^{m} M_{X_i}^j 2^{-j} \tag{10}$$

where $M_{X_i}^j (j = 1, 2, \ldots, m)$ are the digits of mantissa M_{X_i} in base 2. $M_{X_i}^j \in \{0, 1\}$ for $1 \leq j \leq m$ and $1 \leq i \leq N$. Agreeing that $(1/2) \leq M_{X_i}$ implies $M_{X_i}^1$ verifies $M_{X_i}^1 \neq 0$, i.e., $M_{X_i}^1 = 1$.

The exponent E_{X_i} is coded in the form "arithmetic complemented to 2" (necessary to encode the negative exponent)

$$E_{X_i} = \sum_{j=0}^{p-2} E_{X_i}^j - S_{E_{X_i}} 2^{p-1} \tag{11}$$

where $S_{E_{X_i}}$ is the sign bit of E_{X_i} ($S_{E_{X_i}} = 0$ if E_{X_i} is positive or zero, and $S_{E_{X_i}} = 1$ otherwise), and $E_{X_i}^j \in \{0, 1\}$ for $0 \leq j \leq p - 2$ and $1 \leq i \leq N$ (cf. Table II). The exponent E_{X_i}, given by (11), may be calculated by a single neuron (cf. Fig. 4).

We can then represent any element X_i of the input array X as a $(m + p + 1)$ bit binary number, as follows:

$$X_i \equiv \left(S_{X_i}, M_{X_i}^1, M_{X_i}^2, \ldots, M_{X_i}^m, S_{E_{X_i}}, E_{X_i}^0, E_{X_i}^1, \ldots, E_{X_i}^{p-2}\right). \tag{12}$$

Definition 1: Let X_i and X_q be two elements of the input array X. The comparison function of X_i and X_q is defined as follows.

If ($S_{X_q} \neq 1$ and $S_{X_i} \neq 1$ i.e., X_q and X_i are not simultaneously negative), then

$$\text{for } i < q, \quad \text{comp}(X_i, X_q) = \begin{cases} 1, & \text{if } X_q \geq X_i \\ 0, & \text{if } X_q < X_i \end{cases} \tag{13}$$

and

$$\text{for } i > q, \quad \text{comp}(X_i, X_q) = \begin{cases} 1, & \text{if } X_q > X_i \\ 0, & \text{if } X_q \leq X_i. \end{cases} \tag{14}$$

If ($S_{X_q} \neq 0$ and $S_{X_i} \neq 0$, i.e., X_q and X_i are not simultaneously positive), then

$$\text{for } i < q, \quad \text{comp}(X_i, X_q) = \begin{cases} 1, & \text{if } X_q > X_i \\ 0, & \text{if } X_q \leq X_i \end{cases} \tag{15}$$

TABLE II
ARITHMETIC COMPLEMENTED TO TWO

	$S_{E_{X_i}}$	$E_{X_i}^{p-2}$	$E_{X_i}^{p-3}$	...	$E_{X_i}^{1}$	$E_{X_i}^{0}$
Encoding of greatest exponent	0	1	1	...	1	1
Encoding of zero exponent	0	0	0	...	0	0
Encoding of exponent $E_{X_i} = -1$	1	1	1	...	1	1
Encoding of smallest exponent	1	0	0	...	0	0

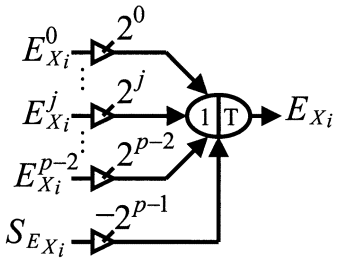


Fig. 4. Neural network enabling calculation of exponent $E_{X_i} (1 \leq i \leq N)$.

and

$$\text{for } i > q, \quad \text{comp}(X_i, X_q) = \begin{cases} 1, & \text{if } X_q \geq X_i \\ 0, & \text{if } X_q < X_i. \end{cases} \tag{16}$$

■

Definition 2: Let X_q be an element of the input array X. The order in the input array X of X_q is defined as

$$\text{ord}(X_q, X) \stackrel{\text{def}}{=} \sum_{i<q} \text{comp}(X_i, X_q) + \sum_{i>q} \text{comp}(X_i, X_q) + 1. \tag{17}$$

Definition 3: Let $X_{(k)}$ denote the kth largest element of the input array X. Let X_q be an element of the input array X. Then

$$X_q = X_{(k)} \quad \text{iff} \quad \text{ord}(X_q, X) = k. \tag{18}$$

■

The task of finding the kth largest element of the input array X can be done in two phases.

1) Compute the order in the input array X of any element $X_q (q = 1, 2, \ldots, N)$.
2) Select and transfer to output the element of the input array X corresponding to the order k desired or chosen by decision-markers (designers).

Note that the operation in either of these two phases can be performed in parallel. This is why it is possible to achieve high processing speed by utilizing the massive parallelism of neural networks.

Corresponding to these two phases, the AOSF is composed of N order networks, a selection network, and an adjustment input, which allows choice of the order k of the element to be transferred to output.

Proposition 1: $\Phi_T(x-1) = x$ if $x \in \{0, 1\}$. ■

Proof: If $x = 0$ then $\Phi_T(x-1) = \Phi_T(-1) = 0$ [cf. function (2)].

If $x = 1$, then $\Phi_T(x-1) = \Phi_T(0) = 1$ [cf. function (2)]. ■

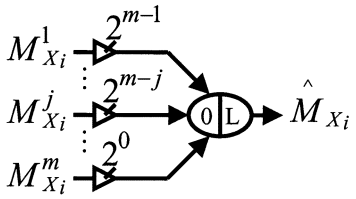


Fig. 5. Neural network enabling calculation of $\widehat{M}_{X_i} (1 \leq i \leq N)$ according to relationship (19).

Let $\widehat{M}_{X_i}$ be the integer associated with $M_{X_i} (i = 1, 2, \ldots, N)$ according to

$$\widehat{M}_{X_i} = 2^m M_{X_i} = \sum_{j=1}^{m} M_{X_i}^j 2^{m-j}. \tag{19}$$

Proposition 2: Let $\vee$ be an operator of the set $\mathcal{R} = \{<, >, \leq, \geq, =\}$. Let $\widehat{M}_{X_i}$ and $\widehat{M}_{X_q}$ be two integers associated to M_{X_i} and M_{X_q}, respectively, according to (19). Then

$$M_{X_i} \vee M_{X_q} \Longleftrightarrow \widehat{M}_{X_i} \vee \widehat{M}_{X_q}. \tag{20}$$

■

Proof: Let $\vee$ be an operator of $\mathcal{R}$.

If $M_{X_i} \vee M_{X_q}$, then $2^m M_{X_i} \vee 2^m M_{X_q} \Rightarrow \widehat{M}_{X_i} \vee \widehat{M}_{X_q}$ [cf. relationship (19)].

If $\widehat{M}_{X_i} \vee \widehat{M}_{X_q}$, then $2^{-m}\widehat{M}_{X_i} \vee 2^{-m}\widehat{M}_{X_q} \Rightarrow M_{X_i} \vee M_{X_q}$ [cf. relationship (19)].

The operator does not change when M_{X_i} and M_{X_q} are multiplied by the positive number 2^m or when $\widehat{M}_{X_i}$ and $\widehat{M}_{X_q}$ are multiplied by the positive number 2^{-m}. ■

The integer $\widehat{M}_{X_i}$ associated with the mantissa M_{X_i} according to (19) is calculated by a linear neuron (cf. Fig. 5), all of whose weights are integers, which facilitate its implementation based on VLSI technology.

Let $|X_i|$ denote the absolute value of X_i; $|X_i|$ is represented by the given couple (M_{X_i}, E_{X_i}) as follows:

$$|X_i| = M_{X_i} 2^{E_{X_i}}. \tag{21}$$

As in (12), $|X_i|$ can be represented as an $(m+p)$ bit binary number

$$|X_i| \equiv \left(M_{X_i}^1, M_{X_i}^2, \ldots, M_{X_i}^m, S_{E_{X_i}}, E_{X_i}^0, E_{X_i}^1, \ldots, E_{X_i}^{p-2}\right). \tag{22}$$

Definition 4: Let $|X_i|$ and $|X_q|$ be the absolute value of X_i and X_q, respectively; we say that $|X_q| > |X_i|$ if and only if

1) $E_{X_q} > E_{X_i}$ or
2) $E_{X_q} = E_{X_i}$ and $M_{X_q} > M_{X_i}$ [or equivalently $\widehat{M}_{X_q} > \widehat{M}_{X_i}$ (cf. Proposition 2)]. ■

Proposition 3: Let $|X_i|$ and $|X_q|$ be the absolute value of X_i and X_q, respectively. Then

$$\operatorname{comp}(|X_i|,|X_q|) = \begin{cases} S_1, & \text{if } i < q \\ S_2, & \text{if } i > q \end{cases} \tag{23}$$

where

$$\begin{aligned} S_1 = \Phi_T \Big[& \Phi_T\left(E_{X_q} - E_{X_i} - 1\right) - \Phi_T\left(E_{X_i} - E_{X_q} - 1\right) \\ & + \Phi_T\Big(- \Phi_T\left(E_{X_q} - E_{X_i} - 1\right) \\ & \quad - \Phi_T\left(E_{X_i} - E_{X_q} - 1\right) \\ & \quad + \Phi_T\left(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1\right) - 1\Big) \\ & - \Phi_T\Big(- \Phi_T\left(E_{X_q} - E_{X_i} - 1\right) \\ & \quad - \Phi_T\left(E_{X_i} - E_{X_q} - 1\right) \\ & \quad + \Phi_T\left(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1\right) - 1\Big) \\ & + \Phi_T\Big(\Phi_T\Big(-\Phi_T\left(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1\right) \\ & \quad - \Phi_T\left(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1\right)\Big) \\ & \quad - \Phi_T\left(E_{X_q} - E_{X_i} - 1\right) \\ & \quad - \Phi_T\left(E_{X_i} - E_{X_q} - 1\right) - 1\Big) \Big] \end{aligned} \tag{24}$$

$$\begin{aligned} S_2 = \Phi_T \Big[& \Phi_T\left(E_{X_q} - E_{X_i} - 1\right) - \Phi_T\left(E_{X_i} - E_{X_q} - 1\right) \\ & + \Phi_T\Big(- \Phi_T\left(E_{X_q} - E_{X_i} - 1\right) - \Phi_T\left(E_{X_i} - E_{X_q} - 1\right) \\ & \quad + \Phi_T\left(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1\right) - 1\Big) \\ & - \Phi_T\Big(- \Phi_T\left(E_{X_q} - E_{X_i} - 1\right) - \Phi_T\left(E_{X_i} - E_{X_q} - 1\right) \\ & \quad + \Phi_T\left(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1\right) - 1\Big) - 1\Big] \end{aligned} \tag{25}$$

and Φ_T is defined in (2). ■

Proof: (See Appendix I.)

The network for computing the comparison function of $|X_i|$ and $|X_q|$ is illustrated by the diagram depicted in Fig. 6. This network is denoted as the absolute value comparison network (AVCN).

Definition 5: Let $|X_i|$ and $|X_q|$ be the absolute value of X_i and X_q, respectively; we say that $X_q > X_i$ if and only if

1) $S_{X_q} = 0$ and $S_{X_i} = 1$ or
2) $S_{X_q} = 0$ and $S_{X_i} = 0$ and $|X_q| > |X_i|$ or
3) $S_{X_q} = 1$ and $S_{X_i} = 1$ and $|X_q| < |X_i|$. ■

Proposition 4: Let $|X_i|$ and $|X_q|$ be the absolute value of X_i and X_q, respectively. Then

$$\begin{aligned} \operatorname{comp}(X_i, X_q) = & \Phi_T\left(-S_{X_q} - S_{X_i} + \operatorname{comp}(|X_i|,|X_q|) - 1\right) \\ & + \Phi_T\left(S_{X_q} + S_{X_i} - \operatorname{comp}(|X_i|,|X_q|) - 2\right) \\ & + \Phi_T\left(-S_{X_q} + S_{X_i} - 1\right). \end{aligned} \tag{26}$$

■

Proof: (See Appendix II.)

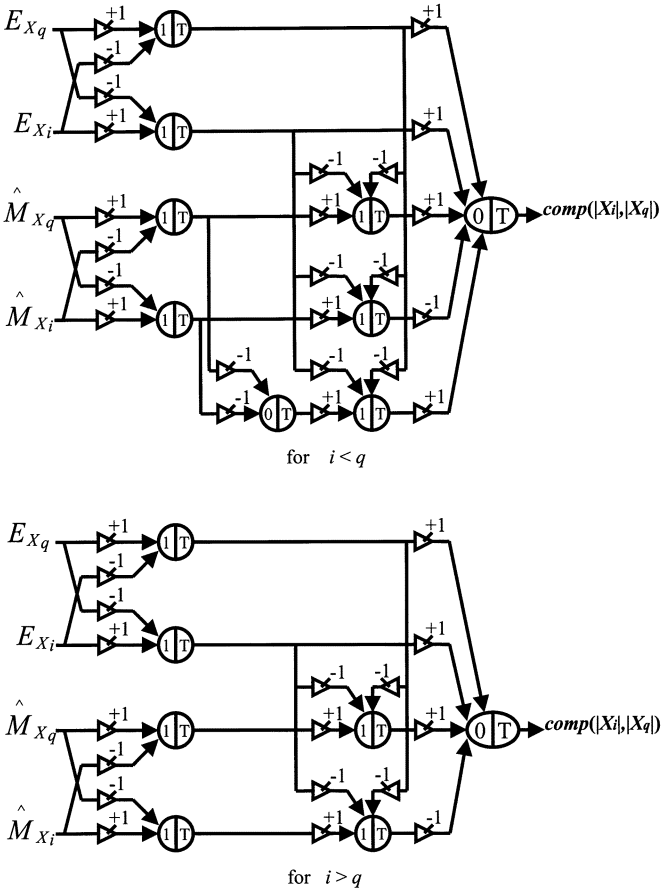


Fig. 6. Neural network, AVCN, for computing the function comparison (17).

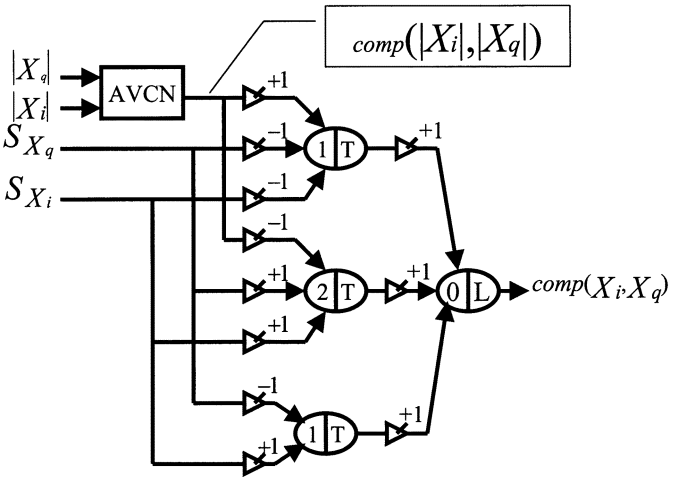


Fig. 7. Comparison network of X_i and X_q, $CN(i,q)$, where AVCN is the network depicted in Fig. 6.

The network for computing the comparison function of X_i and X_q given by (26) is shown in Fig. 7. This network is denoted as $CN(i,q)$ (comparison network of X_i and X_q).

C. Order and Selection Networks

The function of the order network ON_q,$(1 \le q \le N)$ is to compute the order in the input array X of each element X_q (cf. definition 2). The order network ON_q computes the order function (17) and is made up of $(N-1)$ comparison networks,

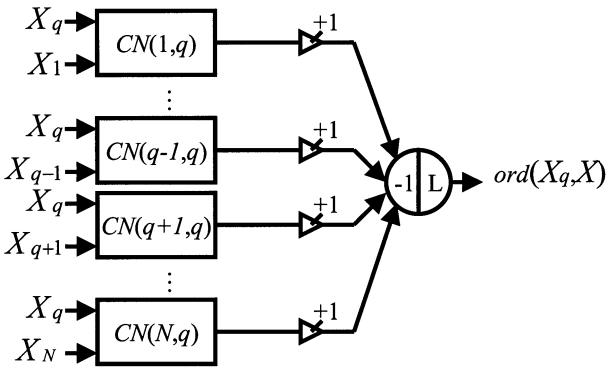


Fig. 8. Order network $ON_q(1 \leq q \leq N)$, where $CN(i,q)$, $i \in \{1,2,\ldots,N\} - \{q\}$ is the network depicted in Fig. 7.

$CN(i,q) i \in \{1,2,\ldots,N\} - \{q\}$, as shown by the diagram in Fig. 8.

The function of the selection network is to select from among the elements of input array X the element corresponding to the order fixed by the adjustment input and to transfer it to output. This network is composed of N equality networks (ENs) and a detection network (DN), which will be studied hereafter.

1) Equality Network: The EN determines whether the order of an element is equal or not to a given number k. The number $k(1 \leq k \leq N)$ is fixed via the adjustment input A_k, according to

$$k = \sum_{q=0}^{n-1} a_{(k)}^q 2^q \tag{27}$$

where $a_{(k)}^0, a_{(k)}^1, \ldots, a_{(k)}^{n-1}$ is the word of command allowing choice of the order of the element to be sent to output. $a_{(k)}^q \in \{0,1\}$ for $0 \leq q \leq n-1$ and $1 \leq k \leq N$.

The function computed by the EN is defined as

$$\mathrm{eq}\left[\mathrm{ord}(X_i,X),k\right] \stackrel{\mathrm{def}}{=} \begin{cases} 0, & \text{if } \mathrm{ord}(X_i,X) = k \\ 1, & \text{otherwise.} \end{cases} \tag{28}$$

Proposition 5:

$$\begin{aligned} \mathrm{eq}\left[\mathrm{ord}(X_i,X),k\right] = {} & \Phi_T\left(\Phi_T\left(\mathrm{ord}(X_i,X) - k - 1\right)\right. \\ & \left. + \Phi_T\left(k - (\mathrm{ord}(X_i,X) - 1) - 1\right)\right. \\ & \text{for } 1 \leq i \leq N \text{ and } 1 \leq k \leq N \end{aligned} \tag{29}$$

where Φ_T is defined in (2). ■

Proof: Three cases can be distinguished.

1) If $\mathrm{ord}(X_i,X) = k$, then $\Phi_T(\mathrm{ord}(X_i,X) - k - 1) = \Phi_T(k - \mathrm{ord}(X_i,X) - 1) = \Phi_T(-1) = 0$ and consequently $\Phi_T(\Phi_T(\mathrm{ord}(X_i,X) - k - 1) + \Phi_T(k - \mathrm{ord}(X_i,X) - 1) - 1) = \Phi_T(-1) = 0$.

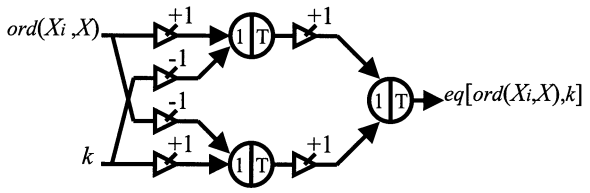


Fig. 9. Equality network EN.

2) If $\mathrm{ord}(X_i,X) < k$, then $\mathrm{ord}(X_i,X) - k \leq -1$ and $k - \mathrm{ord}(X_i,X) \geq 1 \Longrightarrow \Phi_T(\mathrm{ord}(X_i,X) - k - 1) = 0$, $\Phi_T(k - \mathrm{ord}(X_i,X) - 1) = 1$ and $\Phi_T(\mathrm{ord}(X_i,X) - k - 1) + \Phi_T(k - \mathrm{ord}(X_i,X) - 1) - 1 = 0$ and consequently $\Phi_T(\Phi_T(\mathrm{ord}(X_i,X) - k - 1) + \Phi_T(k - \mathrm{ord}(X_i,X) - 1) - 1) = \Phi_T(0) = 1$.
3) If $\mathrm{ord}(X_i,X) > k$, then $\mathrm{ord}(X_i,X) - k \geq 1$ and $k - \mathrm{ord}(X_i,X) \leq -1 \Longrightarrow \Phi_T(\mathrm{ord}(X_i,X) - k - 1) = 1$, $\Phi_T(k - \mathrm{ord}(X_i,X) - 1) = 0$ and $\Phi_T(\mathrm{ord}(X_i,X) - k - 1) + \Phi_T(k - \mathrm{ord}(X_i,X) - 1) - 1 = 0$ and consequently $\Phi_T(\Phi_T(\mathrm{ord}(X_i,X) - k - 1) + \Phi_T(k - \mathrm{ord}(X_i,X) - 1) - 1) = \Phi_T(0) = 1$.

To summarize: $\Phi_T(\Phi_T(\mathrm{ord}(X_i,X) - k - 1) + \Phi_T(k - \mathrm{ord}(X_i,X) - 1) - 1) = 0$ if $\mathrm{ord}(X_i,X) = k$ and $\Phi_T(\Phi_T(\mathrm{ord}(X_i,X) - k - 1) + \Phi_T(k - \mathrm{ord}(X_i,X) - 1) - 1) = 1$, otherwise and consequently $\mathrm{eq}[\mathrm{ord}(X_i,X),k] = \Phi_T(\Phi_T(\mathrm{ord}(X_i,X) - k - 1) + \Phi_T(k - \mathrm{ord}(X_i,X) - 1) - 1)$ [cf. function (28)]. ■

The EN consists of three threshold-logic neurons, as shown by the diagram in Fig. 9.

2) Detection Network: The function of the DN is to detect and send to output the kth largest element $X_{(k)}$ of input array X.

Proposition 6: Let the equation at the bottom of the page [cf. relationship (12)] be the kth largest element of input array X. Then

$$S_{X_{(k)}} = f\left(S_{X_1}, S_{X_2}, \ldots, S_{X_N}\right) \tag{30}$$

$$M_{X_{(k)}}^j = f\left(M_{X_1}^j, M_{X_2}^j, \ldots, M_{X_N}^j\right), \text{ for } 1 \leq j \leq m \tag{31}$$

$$S_{E_{X_{(k)}}} = f\left(S_{E_{X_1}}, S_{E_{X_2}}, \ldots, S_{E_{X_N}}\right) \tag{32}$$

$$E_{X_{(k)}}^q = f\left(E_{X_1}^q, E_{X_2}^q, \ldots, E_{X_N}^q\right), \text{ for } 0 \leq q \leq p-2 \tag{33}$$

where

$$f(b_1, b_2, \ldots, b_N) = \Phi_T\Big(\sum_{i=1}^{N} \Phi_T(b_i - \mathrm{eq}[\mathrm{ord}(X_i,X),k] - 1) - 1\Big)$$

with $bi = (S_{X_i}$ or $M_{X_i}^j (j = 1,2,\ldots,m)$ or $S_{E_{X_i}}$ or $E_{X_i}^q (q = 0,1,\ldots,p-2))$ for $i = 1,2,\ldots,N$. ■

$$X_{(k)} \equiv (S_{X_{(k)}}, M_{X_{(k)}}^1, M_{X_{(k)}}^2, \ldots, M_{X_{(k)}}^m S_{E_{X_{(k)}}}, E_{X_{(k)}}^0, E_{X_{(k)}}^1, \ldots, E_{X_{(k)}}^{p-2})$$

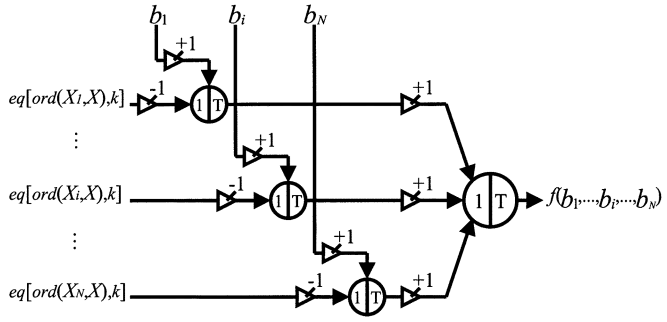


Fig. 10. Detection network DN.

Proof: Suppose that $\mathrm{ord}(X_l, X) = k$, then $\mathrm{eq}[\mathrm{ord}(X_l, X), k] = 0$ and $\mathrm{eq}[\mathrm{ord}(X_i, X), k] = 1 \forall i \in \{1, 2, \ldots, N\} - \{l\}$. Therefore

$$
\begin{aligned}
& f(b_1, b_2, \ldots, b_N) \\
& \quad \stackrel{\mathrm{def}}{=} \Phi_T \left(\sum_{i=1}^{N} \Phi_T \left(b_i - eq\left[ord(X_i, X), k\right] - 1 \right) - 1 \right) \\
& \quad = \Phi_T \left(\Phi_T \left(b_l - eq\left[ord(X_i, X), k\right] - 1 \right) - 1 \right) \\
& \qquad + \Phi_T \left(\sum_{\substack{i \neq 1 \\ 1 \leq i \leq N}}^{N} \Phi_T \left(b_i - eq\left[ord(X_i, X), k\right] - 1 \right) - 1 \right) \\
& \quad = \Phi_T \left(\Phi_T (b_l - 0 - 1) - 1 \right) \\
& \qquad + \Phi_T \left(\sum_{\substack{i \neq 1 \\ 1 \leq i \leq N}}^{N} \Phi_T (b_i - 1 - 1) - 1 \right) \\
& \quad = \Phi_T (b_l - 1) + \Phi_T (0 - 1) \text{ (cf. proposition 1)} \\
& \quad = b_l \text{ [cf. proposition 1 and function (2)]}
\end{aligned}
$$

and consequently

$$
\begin{aligned}
S_{X_l} &= f\left(S_{X_1}, S_{X_2}, \ldots, S_{X_N}\right) \\
M^j_{X_l} &= f\left(M^j_{X_1}, M^j_{X_2}, \ldots, M^j_{X_N}\right), \text{ for } 1 \leq j \leq m \\
S_{E_{X_l}} &= f\left(S_{E_{X_1}}, S_{E_{X_2}}, \ldots, S_{E_{X_N}}\right) \\
E^q_{X_l} &= f\left(E^q_{X_1}, E^q_{X_2}, \ldots, E^q_{X_N}\right), \text{ for } 0 \leq q \leq p-2.
\end{aligned}
$$

As $S_{X_l} = S_{X_{(k)}}$, $M^j_{X_i} = M^j_{X_{(k)}}$ for $1 \leq j \leq m$, $S_{E_{X_l}} = S_{E_{X_{(k)}}}$ and $E^q_{X_l} = E^q_{X_{(k)}}$ for $0 \leq q \leq p-2$, we have

$$
\begin{aligned}
S_{X_{(k)}} &= f\left(S_{X_1}, S_{X_2}, \ldots, S_{X_N}\right) \\
M^j_{X_{(k)}} &= f\left(M^j_{X_1}, M^j_{X_2}, \ldots, M^j_{X_N}\right), \text{ for } 1 \leq j \leq m \\
S_{E_{X_{(k)}}} &= f\left(S_{E_{X_1}}, S_{E_{X_2}}, \ldots, S_{E_{X_N}}\right) \\
E^q_{X_{(k)}} &= f\left(E^q_{X_1}, E^q_{X_2}, \ldots, E^q_{X_N}\right), \text{ for } 0 \leq q \leq p-2.
\end{aligned}
$$

■

Function (30)–(33) are computed by the network shown by the diagram in Fig. 10. The selection network is shown in Fig. 11 and is denoted as *SN*. It transfers only the appropriate element whose order equals k to the output.

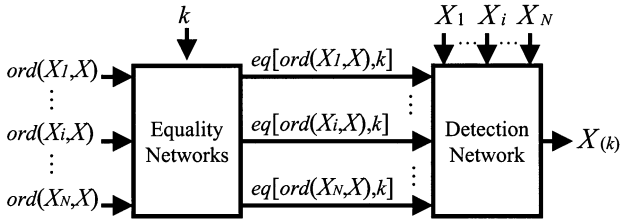


Fig. 11. Selection network SN.

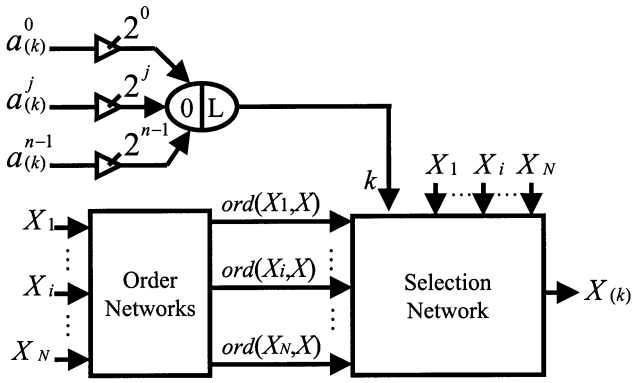


Fig. 12. Adjustable order statistic filter AOSF, where $(a^0_{(k)}, \ldots, a^j_{(k)}, \ldots, a^{n-1}_{(k)})$ is the word of command given by the relationship (27).

D. The AOSF

The AOSF is shown in Fig. 12, where the adjustment input A_k determines which order statistic is to appear at the output. The network illustrated in Fig. 12 consists of two kinds of neurons arranged in 11 layers. The number of neurons in AOSF for input size N is $14N^2+(m+p-9)N+m+p+2$. There are 11 layers of neurons in the AOSF, thus the processing time is 11 times the processing time of a single neuron. As the number of elements of the input array increases, only the number of neurons in each layer increases, not the number of layers themselves. Therefore, AOSF's total processing time remains constant irrespective of the number of element in the input array. This contrasts with conventional hardware implementation of order statistic filters [14], [74], where the processing time increases along with the number of elements.

The claim that the processing speed of AOSF is independent of its input size does not take into account limitations in the hardware implementation. It is based on the assumption that the processing time of a neuron is independent of its input size. This assumption, however, is not true in analog circuits. For instance, as the number of inputs to a neuron increases, the capacitances of the wires that connect these inputs will increase, causing the settling time to the required accuracy to increase. Therefore, the processing speed of the AOSF to some extent depends on the input size. Even with these limitations, however, the processing speed of the AOSF will still be high enough to have the advantage of speed.

Technologies used in AOSF implementation are broadly categorized into silicon [46]–[48], using analog, digital, or

mixed analog/digital integrated circuits, and optical or electrooptical [1], [5]. No matter which medium is used, the performance of the AOSF would inevitably be affected by the current level of the medium's technology. Here, we address some problems that might be faced when the AOSF is implemented using analog VLSI circuits. Such problems are also common to other neural network models; however, because the AOSF has a simple configuration, its implementation is less affected.

The first problem is that of poor absolute accuracy in setting up the values of the connection weights. This problem does not arise if the AOSF is implemented using monolithic analog VLSI circuits.

Whatever technology is utilized, the AOSF is not affected by this problem, since the AOSF has a very simple configuration, its weights are all fixed, and most of them are just $+1$ or -1; they can be set simply by connecting the input to the neuron or by inverting the input before connection.

The second problem is due to the saturation characteristics of the amplifiers used in implementing the linear neuron. For some practical applications, this may not be a serious problem. For example, in image processing applications, the input to the AOSF can be easily scaled to fit in the linear range of the amplifiers.

III. Applications

This section presents important examples of the extension of the AOSF to sorting and to adaptive order statistic filters. Finally, an important application of minimum filter for implementing a special neural network model, the NNC, is described.

A. Sorting

Sorting has many applications, especially in data analysis and image processing [80]. Sorting an array is equivalent to giving all order statistics of the array and arranging them either in ascending or descending order.

A more efficient implementation of the sorting network is shown in the diagram in Fig. 13(a). This sorting network is equivalent to N AOSFs set up in parallel, whose common module "order networks" has been merged.

Sorting time is fixed and is only 11 times the processing time for a single neuron. Merging the common module "order networks" permits considerable reduction of the size of the sorting network and a gain of approximately N^3 neurons. A detailed account of a similar implementation can be found in [49]–[51].

A second implementation of the sorting network consists of using n separate AOSF networks in parallel, as shown in the diagram in Fig. 13(b). Sorting time is fixed and does not depend on the size of the input; it is the same time taken for processing a single AOSF.

A third implementation is the use of a single AOSF network [cf. Fig. 13(c)]. By changing the value of k from 1 to N, the elements of the sorted array will appear at the output sequentially. The advantage here is that less neurons are needed; the disadvantage is that the sorting time is proportional to the size of the input.

At each clock pulse [cf. Fig. 13(c)], the counter changes state, and k goes from one value to the next. The AOSF finally produces at its output the input array element whose order corresponds to new value of k. The clock frequency must be lower than AOSF processing speed, i.e., lower than $(1/(11\tau))$, where τ is the processing time for a single neuron.

B. Adaptive Order Statistic Filters

This subsection presents an example of implementing in fixed time of a type of adaptive order statistic filter called comparison-and-selection (CS) filter [38]. The output of the CS filter with parameter J at position l for the input $X_l = (x_{l-s}, \ldots, x_l, \ldots, x_{l+s})$ is defined as

$$y_l = \begin{cases} x_l^{(s+1+J)}, & \text{if } u_l \geq x_l^{(s+1)} \\ x_l^{(s+1+J)}, & \text{otherwise} \end{cases} \tag{34}$$

where $x_l^{(i)}$ is the ith largest element in the array X_l, u_l and $x_l^{(s+1)}$ are the sample mean and median, respectively, and J is an integer satisfying $1 \leq J \leq N$.

Implementation of the CS filter necessitates the calculation of the sample mean u_l according to

$$u_l = \frac{1}{2s+1} \sum_{j=l-s}^{l+s} x_j \tag{35}$$

where $x_j = S_{x_j} M_{x_j} 2^{E_{x_j}}$, as in relationship (9).

Proposition 7: Let E_{x_j} be the exponent associated with $x_j (j = l-s, \ldots, l, \ldots, l+s)$ according to (11). The expression $2^{E_{x_j}}$ can be evaluated as

$$\begin{aligned} 2^{E_{x_j}} = {} & \left(1 - S_{E_{x_j}}\right) \left(\sum_{i=1}^{2^{p-1}-1} \Phi_T \left(E_{xj} - i\right) 2^{i-1} + 1 \right) \\ & + S_{E_{x_j}} \left(1 - \sum_{i=1}^{2^{p-1}} \Phi_T \left(-E_{xj} - i\right) 2^{i} \right) \end{aligned} \tag{36}$$

where Φ_T is defined in (2). ■

Proof:

1) Suppose $S_{E_{x_j}} = 0$ (i.e., $E_{x_j} \geq 0$); then

$$\begin{aligned} & \left(1 - S_{E_{x_j}}\right) \left(\sum_{i=1}^{2^{p-1}-1} \Phi_T \left(E_{xj} - i\right) 2^{i-1} + 1 \right) \\ & \quad + S_{E_{x_j}} \left(1 - \sum_{i=1}^{2^{p-1}} \Phi_T \left(-E_{xj} - i\right) 2^{-i} \right) \\ & \quad = \sum_{i=1}^{2^{p-1}-1} \Phi_T \left(E_{xj} - i\right) 2^{i-1} + 1. \end{aligned}$$

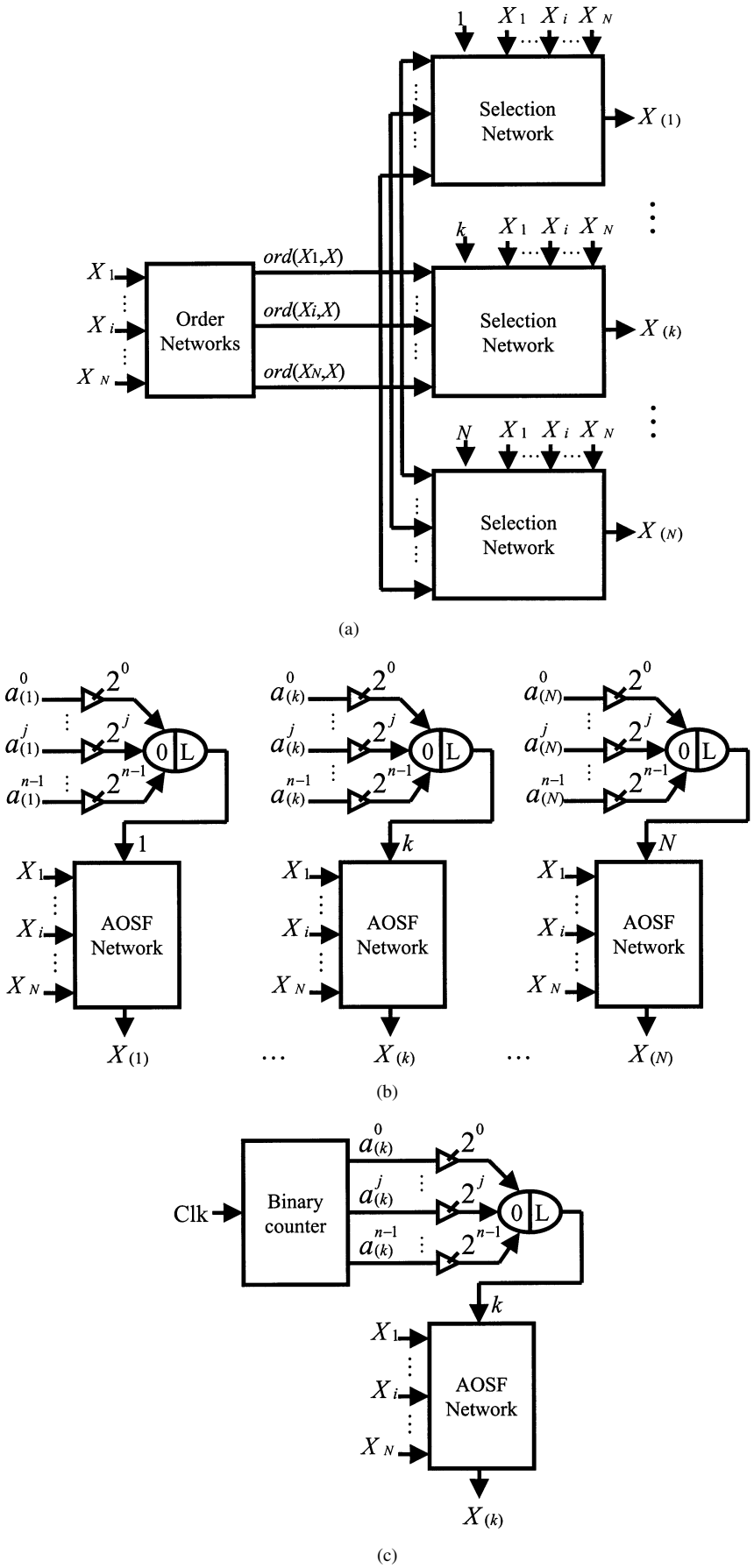


Fig. 13. (a) Sorting network made up of N AOSF networks in parallel whose common module "order networks" has been merged. (b) Sorting network made up of N separate AOSF networks in parallel. (c) Sequential sorting network made up of a single AOSF network.

Two cases can be distinguished.

(a) $E_{x_j} = 0$, then $\Phi_T(E_{x_j} - i) = 0 \ \forall i \in \{1, 2, \ldots, 2^{p-1} - 1\}$, and consequently

$$\sum_{i=1}^{2^{p-1}-1} \Phi_T\left(E_{xj} - i\right) 2^{i-1} + 1 = \sum_{i=1}^{2^{p-1}-1} 0 \times 2^{i-1} + 1 = 1 = 2^{E_{xj}=0}.$$

(b) $E_{x_j} > 0$, then $\Phi_T(E_{x_j} - i) = 0 \ \forall i > E_{x_j}$, and $\Phi_T(E_{x_j} - i) = 1$ otherwise. Therefore

$$\begin{aligned}
&\sum_{i=1}^{2^{p-1}-1} \Phi_T\left(E_{xj} - i\right) 2^{i-1} + 1 \\
&= \sum_{i=1}^{E_{xj}} \Phi_T\left(E_{xj} - i\right) 2^{i-1} + \sum_{i=E_{xj}+1}^{2^{p-1}-1} \Phi_T\left(E_{xj} - i\right) 2^{i-1} + 1 \\
&= \sum_{i=1}^{E_{xj}} 2^{i-1} + \sum_{i=E_{xj}+1}^{2^{p-1}-1} 0 \times 2^{i-1} + 1 \\
&= \sum_{i=1}^{E_{xj}} 2^{i-1} + 1 \\
&= 2^{E_{xj}} - 1 + 1 \\
&= 2^{E_{xj}}.
\end{aligned}$$

2) Suppose $S_{E_{x_j}} = 1$ (i.e., $E_{x_j} < 0$); then

$$\begin{aligned}
&\left(1 - S_{E_{xj}}\right)\left(\sum_{i=1}^{2^{p-1}-1} \Phi_T\left(E_{xj} - i\right) 2^{i-1} + 1\right) \\
&\quad + S_{E_{xj}}\left(1 - \sum_{i=1}^{2^{p-1}} \Phi_T\left(-E_{xj} - i\right) 2^{-i}\right) \\
&\qquad = 1 - \sum_{i=1}^{2^{p-1}} \Phi_T\left(-E_{xj} - i\right) 2^{-i} \\
&\qquad = 1 - \sum_{i=1}^{-E_{xj}} \Phi_T\left(-E_{xj} - i\right) 2^{-i} \\
&\qquad\quad - \sum_{i=-E_{xj}+1}^{2^{p-1}-1} \Phi_T\left(-E_{xj} - i\right) 2^{-i} \\
&\qquad = 1 - \sum_{i=1}^{-E_{xj}} 2^{-i} - \sum_{i=-E_{xj}+1}^{2^{p-1}} 0 \times 2^{-i} \\
&\qquad = 1 - \left(1 - 2^{E_{xj}}\right) \\
&\qquad = 2^{E_{xj}}.
\end{aligned}$$

■

The expression $2^{E_{x_j}}$ can be calculated by the network illustrated in Fig. 14. The difference between the sample mean u_l and the median $x_l^{(s+1)}$, $u_l - x_l^{(s+1)} = (1/(2s+1)) \sum_{q=l-s}^{l+s} x_q - x_l^{(s+1)}$ can be calculated by the network illustrated in Fig. 15.

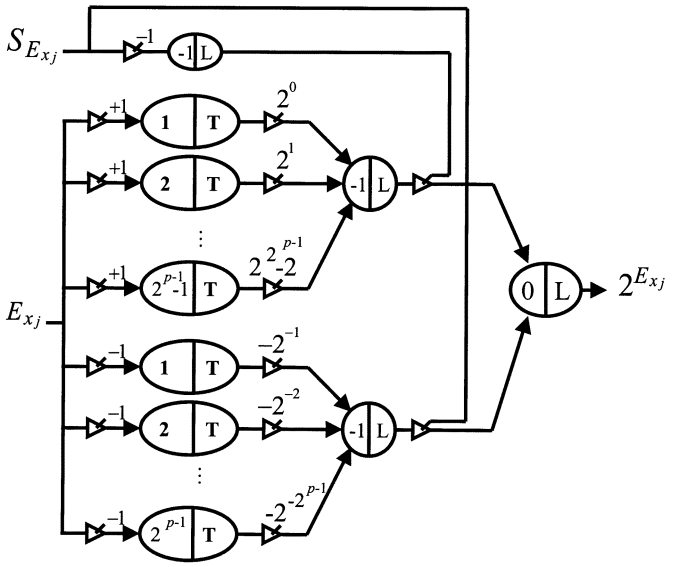


Fig. 14. Network for calculating the expression $2^{E_{x_j}}$.

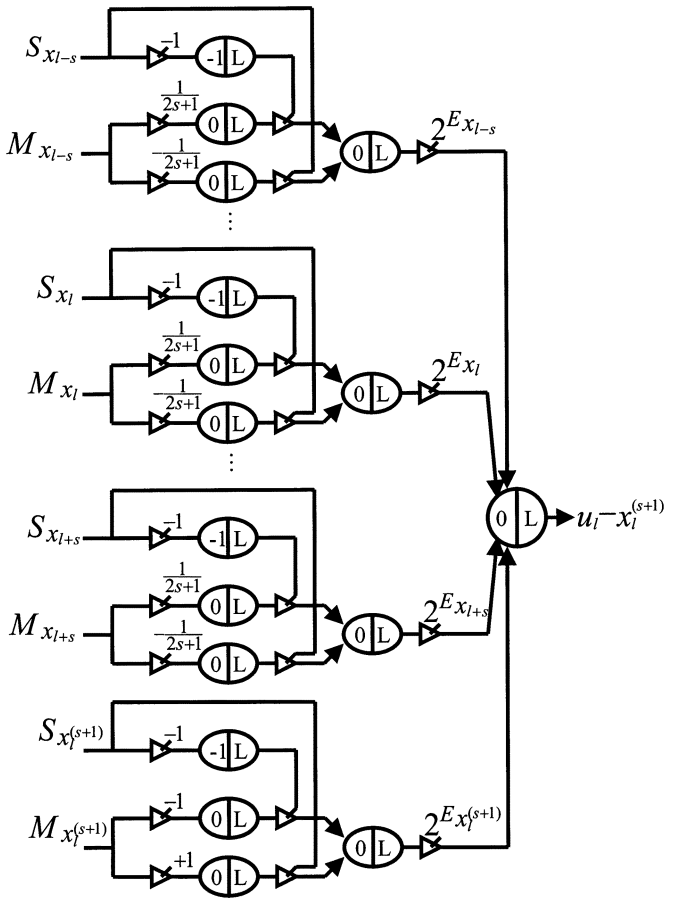


Fig. 15. Network for calculating the difference between u_l and $x_l^{(s+1)}$, $u_l - x_l^{(s+1)} = (1/(2s+1)) \sum_{q=l-s}^{l+s} x_q - x_l^{(s+1)}$.

Proposition 8: The function (34) can be evaluated as

$$y_l = x_l^{(\mu)} \tag{37}$$

where

$$\mu = s + 1 + \left(2 \times \Phi_T\left(u_l - x_l^{(s+1)}\right) - 1\right) \times J \tag{38}$$

and Φ_T is defined in (2). ■

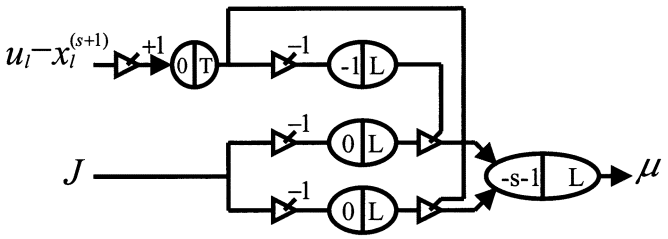


Fig. 16. Network for calculating the parameter μ defined in Proposition 8 by relationship (37).

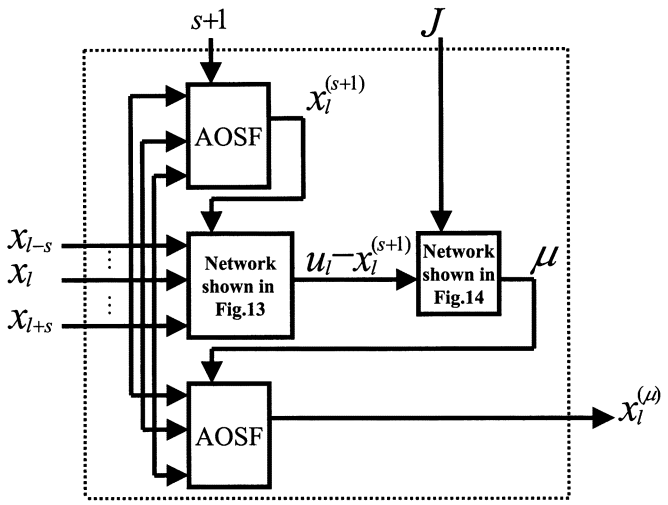


Fig. 17. Neural network implementation of the CS filter.

Proof: If $u_l \geq x_l^{(s+1)}$, then $\Phi_T(u_l - x_l^{(s+1)}) = 1$ and $\mu = s + 1 + J$. Therefore

$$y_l = x_l^{(s+1+J)}.$$

If $u_l < x_l^{(s+1)}$, then $\Phi_T(u_l - x_l^{(s+1)}) = 0$ and $\mu = s + 1 - J$. Therefore

$$y_l = x_l^{(s+1-J)}.$$

To summarize: $y_l = x_l^{(s+1+J)}$ if $u_l \geq x_l^{(s+1)}$, and $y_l = x_l^{(s+1-J)}$ otherwise. ■

The parameter μ defined in Proposition 8 by relationship (38) can be calculated by the network illustrated in Fig. 16.

The neural network implementation of the CS filter is shown in Fig. 17, where $u_l - x_l^{(s+1)}$ and μ are calculated by the networks depicted in Figs. 15 and 16, respectively.

C. Neural Network Classifier

In this subsection, our efforts will center on the hardware design of the NNC. First, however, the MDC technique is briefly described.

Consider the classification problem of l classes $C_1, C_2, \ldots C_l$ where each pattern class C_j has a reference or template pattern T^j. An MDC scheme with respect to $T^1, T^2, \ldots, T^l$ classifies the unknown pattern X to class C_j if $d(X, T^j) = \min_{1 \leq i \leq l}\{d(X, T^i)\}$, where $d(X, T^j)$ is the distance defined between X and T^j. In statistical pattern recognition, X is a feature vector; in syntactic pattern recognition, X is a structure such as a string, tree, or graph.

Let $X = (x_1, x_2, \ldots, x_K)$ and $T^j = (t_1^j, t_2^j, \ldots, t_K^j)$ be two points in feature space. The distance measure between X and T^j is defined as

$$d(X, T^j) = K - \sum_{i=1}^{K} dif\left(x_i, t_i^j\right) \tag{39}$$

where

$$dif\left(x_i, t_i^j\right) = \begin{cases} 1, & \text{if } x_i = t_i^j \\ 0, & \text{otherwise.} \end{cases} \tag{40}$$

Proposition 9: Let α and β be two integers, as follows:

$$dif(\alpha, \beta) = \Phi_T\left(-\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1)\right) \tag{41}$$

where Φ_T is defined in (2). ■

Proof: Three cases can be distinguished.

1) If $\alpha = \beta$, then $\Phi_T(\alpha - \beta - 1) = \Phi_T(\beta - \alpha - 1) = \Phi_T(-1) = 0$ and $\Phi_T(-\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1)) = \Phi_T(0) = 1$.
2) If $\alpha < \beta$, then $\alpha - \beta \leq -1$ and $\beta - \alpha \geq 1 \Rightarrow \Phi_T(\alpha - \beta - 1) = 0$, $\Phi_T(\beta - \alpha - 1) = 1$, and $\Phi_T(-\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1)) = \Phi_T(-1) = 0$.
3) If $\alpha > \beta$, then $\alpha - \beta \geq 1$ and $\beta - \alpha \leq -1 \Rightarrow \Phi_T(\alpha - \beta - 1) = 1$, $\Phi_T(\beta - \alpha - 1) = 0$ and $\Phi_T(-\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1)) = \Phi_T(-1) = 0$.

To summarize: If $\alpha = \beta$, then $\Phi_T(-\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1)) = 1$, otherwise $\Phi_T(-\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1)) = 0$. Therefore

$$\mathrm{dif}(\alpha, \beta) = \Phi_T\left(-\Phi_T(\alpha - \beta - 1) - \Phi_T(\beta - \alpha - 1)\right).$$

■

Functions (41) and (40) may be calculated by the networks illustrated in Figs. 18 and 19, respectively. These networks are denoted as BDN (the basic difference network) and $DN(x_i, y_i)$ (the difference network of x_i and y_i), respectively.

Proposition 10: Let $X = (x_1, x_2, \ldots, x_K)$ and $Y = (y_1, y_2, \ldots, y_K)$ be two elements in K-dimensional space. Let $(S_{x_i}, M_{x_i}, E_{x_i})$ and $(S_{y_i}, M_{y_i}, E_{y_i})$ be respective representations of x_i and y_i [cf. relationship (9)], as follows:

$$\text{For all } 1 \leq i \leq K, \quad \mathrm{dif}(x_i, y_i) = \Phi_T\left(\mathrm{dif}\left(S_{x_i}, S_{y_i}\right) + \mathrm{dif} \times \left(\widehat{M}_{x_i}, \widehat{M}_{y_i}\right) + \mathrm{dif}\left(E_{x_i}, E_{y_i}\right) - 3\right) \tag{42}$$

where $\widehat{M}_{x_i}$ and $\widehat{M}_{y_i}$ are the integers associated with M_{x_i} and M_{y_i} according to formula (19), and Φ_T is defined in (2). ■

Proof: $\Phi_T(\mathrm{dif}(S_{x_i}, S_{y_i}) + \mathrm{dif}(\widehat{M}_{x_i}, \widehat{M}_{y_i}) + \mathrm{dif}(E_{x_i}, E_{y_i}) - 3) = 1$ iff $(\mathrm{dif}(S_{x_i}, S_{y_i}) = 1$ and $\mathrm{dif}(\widehat{M}_{x_i}, \widehat{M}_{y_i}) = 1$ and $\mathrm{dif}(E_{x_i}, E_{y_i}) = 1)$. We know the following:

- $\mathrm{dif}(S_{x_i}, S_{y_i}) = 1$ iff $S_{x_i} = S_{y_i}$.
- $\mathrm{dif}(\widehat{M}_{x_i}, \widehat{M}_{y_i}) = 1$ iff $\widehat{M}_{x_i} = \widehat{M}_{y_i}$, or equivalently $M_{x_i} = M_{y_i}$ (cf. Proposition 2).
- $\mathrm{dif}(E_{x_i}, E_{y_i}) = 1$ iff $E_{x_i} = E_{y_i}$.

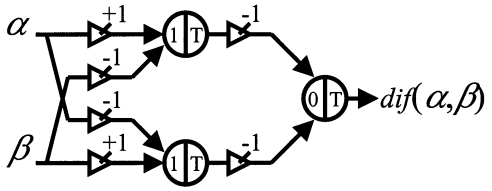


Fig. 18. Basic difference network.

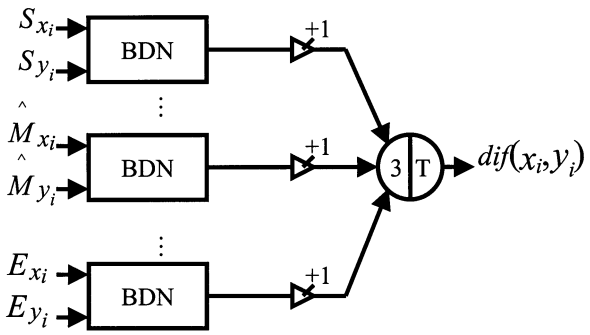


Fig. 19. Difference network of x_i and y_i, $DN(x_i, y_i)$, enabling calculation of expression (40), where BDN is the network shown in Fig. 18.

To summarize

$$\Phi_T\left(\text{dif}\,(S_{x_i},S_{y_i})+\text{dif}\left(\widehat{M}_{x_i},\widehat{M}_{y_i}\right)+\text{dif}\,(E_{x_i},E_{y_i})-3\right) = 1 \text{ iff } (S_{x_i}=S_{y_i} \text{ and } M_{x_i}=M_{y_i} \text{ and } E_{x_i}=E_{y_i})$$

or again

$$\Phi_T\left(\text{dif}\,(S_{x_i},S_{y_i})+\text{dif}\left(\widehat{M}_{x_i},\widehat{M}_{y_i}\right)+\text{dif}\,(E_{x_i},E_{y_i})-3\right) = 1 \text{ iff } x_i=y_i$$

and consequently

$$\text{dif}(x_i,y_i)=\Phi_T\left(\text{dif}\,(S_{x_i},S_{y_i}) +\text{dif}\left(\widehat{M}_{x_i},\widehat{M}_{y_i}\right)+\text{dif}\,(E_{x_i},E_{y_i})-3\right).$$

■

Function (39) is computed by the distance evaluation network (DEN) illustrated by the diagram in Fig. 20.

The classifier NNC is composed of l DENs already seen above (cf. Fig. 20) and a transference network, which will be studied subsequently. For the NNC, l distances must be calculated between the input feature vector X and l reference vectors $T^1, T^2, \ldots, T^l$, according to (39) and (40). The minimum distance among the l distances to the input feature vector X, $\min_{1\le i\le l}\{d(X,T^i)\}$ must be selected. The index of the reference vector, whose distance to the input feature vector X is equal to $\min_{1\le i\le l}\{d(X,T^i)\}$, is produced as the class of X, as follows:

$$\text{class}(X)=q \text{ if } d(X,T^q)=\min_{1\le i\le l}\left\{d(X,T^i)\right\}. \quad (43)$$

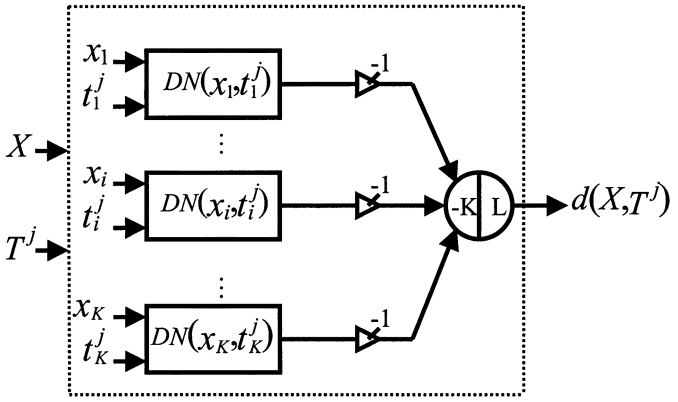


Fig. 20. Distance evaluation network.

Proposition 11: Let $X=(x_1,x_2,\ldots,x_K)$ and $T^j=(T_1^j,T_2^j,\ldots,T_K^j)$ be the input feature and the jth reference vectors, respectively. The class of X can be evaluated based on

$$\text{class}(X)=\sum_{p=1}^{l}\text{dif}\left[d(X,T^p),\min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].p. \quad (44)$$

■

Proof: Suppose $d(X,T^q)=\min_{1\le i\le l}\{d(X,T^i)\}$, then $\text{class}(X)=q$ [cf. (43)], $\text{dif}[d(X,T^q),\min_{1\le i\le l}\{d(X,T^i)\}]=1$ and $\text{dif}[d(X,T^p),\min_{1\le i\le l}\{d(X,T^i)\}]=0\ \forall p\in\{1,2,\ldots,l\}-\{q\}$. Therefore

$$\begin{aligned}\sum_{p=1}^{l}\text{dif}\left[d(X,T^p),\min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].p &= \text{dif}\left[d(X,T^q),\min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].q \\ &\quad+\sum_{\substack{p\ne q\\ 1\le p\le l}}\text{dif}\left[d(X,T^p),\min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].p \\ &= 1\times q+\sum_{\substack{p\ne q\\ 1\le p\le l}}0\times p \\ &= q.\end{aligned}$$

■

Proposition 12: Let $X=(x_1,x_2,\ldots,x_K)$ and $T^p=(t_1^p,t_2^p,\ldots,t_K^p)$ be the input and the pth reference vectors, respectively, where $p\in\{1,2,\ldots,l\}$, as follows:

$$d(X,T^q)=\min_{1\le i\le l}\left\{d(X,T^i)\right\}\Longleftrightarrow\forall j\in\{1,2,\ldots,K\},\ \sum_{p=1}^{l}\text{dif}\left[d(X,T^p),\min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].t_j^p=t_j^q.$$

■

Proof:

1) Suppose $d(X,T^q)=\min_{1\le i\le l}\{d(X,T^i)\}$, then $\text{dif}[d(X,T^q),\min_{1\le i\le l}\{d(X,T^i)\}]=1$ and

$\mathrm{dif}[d(X,T^p), \min_{1\le i\le l}\{d(X,T^i)\}] = 0 \ \forall p \in \{1,2,\ldots,l\} - \{q\}$. Therefore

$$\begin{aligned}
&\forall j \in \{1,2,\ldots,K\},\\
&\sum_{p=1}^{l} \mathrm{dif}\left[d(X,T^p), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].t_j^p\\
&\quad = \mathrm{dif}\left[d(X,T^q), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].t_j^q\\
&\qquad + \sum_{\substack{p\neq q\\ 1\le p\le l}} \mathrm{dif}\left[d(X,T^p), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].t_j^p\\
&\quad = 1\times t_j^q + \sum_{\substack{p\neq q\\ 1\le p\le l}} 0\times t_j^p\\
&\quad = t_j^q.
\end{aligned}$$

2) Suppose $\forall j \in \{1,2,\ldots,K\}$, $\sum_{p=1}^{l}\mathrm{dif}[d(X,T^p), \min_{1\le i\le l}\{d(X,T^i)\}].t_j^p = t_j^q$.

Taking into account the fact that $\sum_{p=1}^{l}\mathrm{dif}[d(X,T^p), \min_{1\le i\le l}\{d(X,T^i)\}] = 1$ and $\forall p \in \{1,2,\ldots,l\}$, $\mathrm{dif}[d(X,T^p), \min_{1\le i\le l}\{d(X,T^i)\}] \in \{0,1\}$, we have the equation shown at bottom of page.

As $\sum_{p=1}^{l}\mathrm{dif}[d(X,T^p), \min_{1\le i\le l}\{d(X,T^i)\}] = 1$, we then have

$$\mathrm{dif}\left[d(X,T^q), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right] = 1$$

and then

$$d(X,T^q) = \min_{1\le i\le l}\left\{d(X,T^i)\right\}.$$

■

The function of the transference network is to select and transfer to output both the class assigned to the input feature vector X and the reference vector $\widehat{T} \in \{T^1,T^2,\ldots,T^l\}$, which satisfies

$$d(X,\widehat{T}) = \min_{1\le i\le l}\left\{d(X,T^i)\right\}. \tag{45}$$

The transference network (TN) is built by combining a minimum filter (i.e., AOSF for $k = N$), l DNs, and $K+1$ linear neurons, as shown in Fig. 21. The minimum filter taking as input the l distances to the input feature vector X, $d(X,T^i)$, $i = 1,2,\ldots,l$, selects the minimum distance $\min_{1\le i\le l}\{d(X,T^i)\}$. The minimum distance $\min_{1\le i\le l}\{d(X,T^i)\}$ thus calculated by the minimum filter is used by the l DNs and $K+1$ linear neurons to identify and transfer to output both the class assigned to the input feature vector X and the reference vector $\widehat{T}$, which satisfies relationship (45).

The network NNC illustrated in Fig. 22 is constructed out of two types of neurons, threshold-logic and linear neurons, arranged in 19 layers. Total processing time is constant, irrespective of the number of reference vectors l and the dimension of the feature vectors K and is only 19 times that of a single neuron, in contrast to conventional hardware implementation, where the processing time for classifying an unknown vector is proportional to $(l \times K)$. Automatic recognition of handwritten numerals and characters has been an active subject of research due to its potential for intelligent man–machine interface, [2], [16], [22], [35]–[37], [39], [40], [45], [57], [60], [64], [70], [72], [75]. Handwritten numeral and character recognition has been computed via either statistical or syntactic approaches. In the statistical approach, a pattern is represented by a set of K-dimensional feature vectors, and the decision-making

$$\begin{aligned}
&\forall j \in \{1,2,\ldots,K\}, \sum_{p=1}^{l}\mathrm{dif}\left[d(X,T^p), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].t_j^p = \left(\sum_{p=1}^{l}\mathrm{dif}\left[d(X,T^p), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right]\right).t_j^q\\
&\quad\Longleftrightarrow \forall j \in \{1,2,\ldots,K\}, \sum_{\substack{p\neq q\\ 1\le p\le l}} \mathrm{dif}\left[d(X,T^p), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].t_j^p\\
&\qquad + \left(\mathrm{dif}\left[d(X,T^q), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right] - \sum_{p=1}^{l}\mathrm{dif}\left[d(X,T^p), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right]\right).t_j^q = 0\\
&\quad\Longleftrightarrow \forall j \in \{1,2,\ldots,K\}, \sum_{\substack{p\neq q\\ 1\le p\le l}} \mathrm{dif}\left[d(X,T^p), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].t_j^p\\
&\qquad - \sum_{\substack{p\neq q\\ 1\le p\le l}}^{l} \mathrm{dif}\left[d(X,T^p), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].t_j^q = 0\\
&\quad\Longleftrightarrow \forall j \in \{1,2,\ldots,K\}, \sum_{\substack{p\neq q\\ 1\le p\le l}} \mathrm{dif}\left[d(X,T^p), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right].\left(t_j^p - t_j^q\right) = 0\\
&\quad\Longleftrightarrow \forall j \in \{1,2,\ldots,l\} - \{q\}, \quad \mathrm{dif}\left[d(X,T^p), \min_{1\le i\le l}\left\{d(X,T^i)\right\}\right] = 0.
\end{aligned}$$

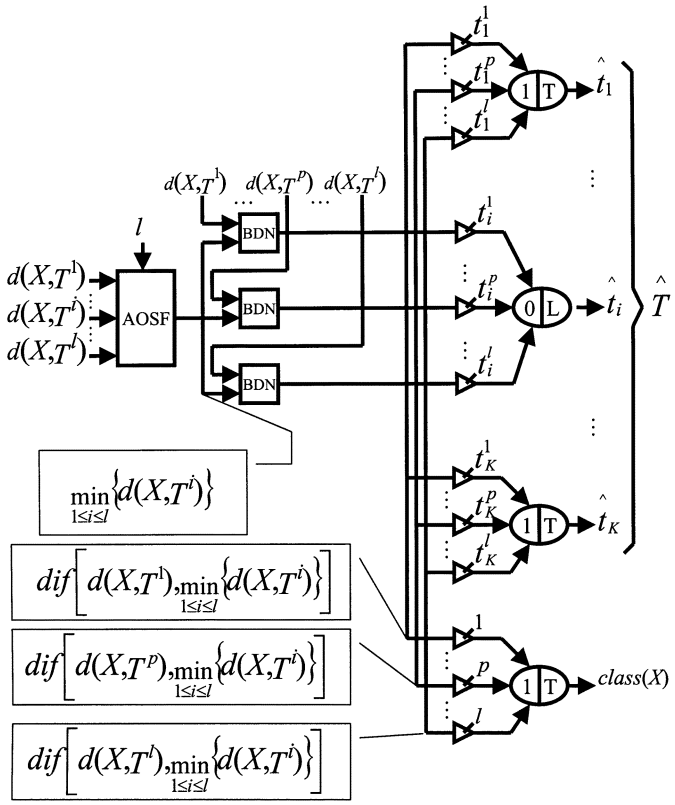


Fig. 21. Transference network.

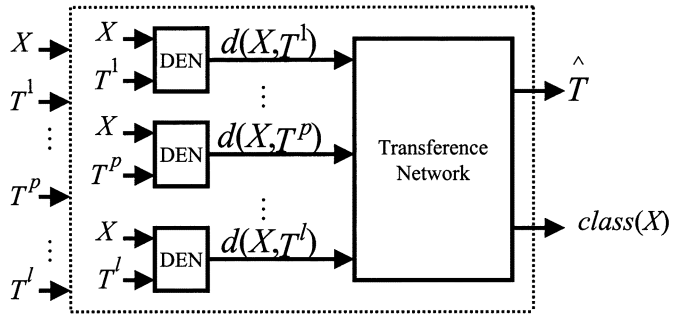


Fig. 22. Neural network classifier.

process is determined by a similarity measure such as a distance metric or a discrimant function. NNC can thus be applied very beneficially to the difficult real-world problem of handwritten numeral and character recognition. NNC is computationaly attractive when compared with a conventional pattern classifier [78].

IV. Conclusion

We have shown a neural network implementation in fixed time of adjustable order statistic filter AOSF. The AOSF is used as a building block for implementing in fixed time all members of the OSF family, including sorting and adaptive order statistic filters. An application of AOSF (for $k = N$) for implementing in fixed time a special neural network model NNC is given.

All neural networks herein proposed have a feed-forward structure and consist of two kinds of neurons—linear and threshold-logic neurons. Among all the neurons proposed in the literature, they are probably the easiest to implement in hardware. These neural networks have a very simple configuration, which makes hardware implementation less subject to any problems caused by poor absolute accuracy in setting up the values of the connection weights. Furthermore, these neural networks' architecture is regular and simple: the connection strengths between the neurons are all fixed, and most of them are just $+1$ or -1. Therefore, this will greatly facilitate actual hardware implementation of proposed neural networks using currently available VLSI technology.

Appendix I
Proof of Proposition 3

- For $i < q$
  1) $E_{x_q} > E_{x_i} \Longrightarrow E_{x_q} - E_{x_i} \geq 1$ and $E_{x_i} - E_{x_q} \geq -1$ (as E_{x_q} and E_{x_i} are integers) $\Longrightarrow \Phi_T(E_{X_q} - E_{X_i} - 1) = 1$ and $\Phi_T(E_{X_i} - E_{X_q} - 1) = 0$ [cf. function (2)],
     which leads to
     $\Phi_T(-\Phi_T(E_{X_q} - E_{X_i} - 1) - \Phi_T(E_{X_i} - E_{X_q} - 1) + \Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - 1) = \Phi_T(\Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - 2) = 0$ (as $\Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - 2 \leq -1$), $\Phi_T(-\Phi_T(E_{X_q} - E_{X_i} - 1) - \Phi_T(E_{X_i} - E_{X_q} - 1) + \Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1) - 1) = \Phi_T(\Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1) - 2) = 0$ (as $\Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1) - 2 \leq -1$)) and $\Phi_T(\Phi_T(-\Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - \Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1)) - \Phi_T(E_{X_q} - E_{X_i} - 1) - \Phi_T(E_{X_i} - E_{X_q} - 1) - 1) = \Phi_T(\Phi_T(-\Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - \Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1)) - 2) = 0$ (as $\Phi_T(-\Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - \Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1)) - 2 \leq -1$),
     and consequently
     $S_i = \Phi_T(1) = 1$ [cf. function (2)];
  2) $E_{x_q} < E_{x_i} \Longrightarrow E_{x_i} - E_{x_q} \geq 1$ and $E_{x_q} - E_{x_i} \leq -1$ (as E_{x_q} and E_{x_i} are integers) $\Longrightarrow \Phi_T(E_{X_q} - E_{X_i} - 1) = 0$ and $\Phi_T(E_{X_i} - E_{X_q} - 1) = 1$ [cf. function (2)],
     which leads to
     $\Phi_T(-\Phi_T(E_{X_q} - E_{X_i} - 1) - \Phi_T(E_{X_i} - E_{X_q} - 1) + \Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - 1) = \Phi_T(\Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - 2) = 0$ (as $\Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - 2 \leq -1$), $\Phi_T(-\Phi_T(E_{X_q} - E_{X_i} - 1) - \Phi_T(E_{X_i} - E_{X_q} - 1) + \Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1) - 1) = \Phi_T(\Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1) - 2) = 0$ (as $\Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1) - 2 \leq -1$)) and $\Phi_T(\Phi_T(-\Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - \Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1)) - \Phi_T(E_{X_q} - E_{X_i} - 1) - \Phi_T(E_{X_i} - E_{X_q} - 1) - 1) = \Phi_T(\Phi_T(-\Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - \Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1)) - 2) = 0$ (as $\Phi_T(-\Phi_T(\widehat{M}_{X_q} - \widehat{M}_{X_i} - 1) - \Phi_T(\widehat{M}_{X_i} - \widehat{M}_{X_q} - 1)) - 2 \leq -1$),
     and consequently
     $S_i = \Phi_T(-1) = 0$ [cf. function (2)];
  3) $E_{x_q} = E_{x_i} \Longrightarrow E_{x_i} - E_{x_q} - 1 = E_{X_q} - E_{X_i} - 1 = -1 \Longrightarrow \Phi_T(E_{x_q} - E_{x_i} - 1) = \Phi_T(E_{X_i} - E_{X_q} - 1) = 0$ [cf. function (2)],

which leads to

$\Phi_T(-\Phi_T(E_{X_q}-E_{X_i}-1)-\Phi_T(E_{X_i}-E_{X_q}-1)+\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-1)=\Phi_T(\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-1)=\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)$ (cf. Proposition 1), $\Phi_T(-\Phi_T(E_{X_q}-E_{X_i}-1)-\Phi_T(E_{X_i}-E_{X_q}-1)+\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-1)=\Phi_T(\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-1)=\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)$ (cf. Proposition 1), and $\Phi_T(\Phi_T(-\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1))-\Phi_T(E_{X_q}-E_{X_i}-1)-\Phi_T(E_{X_i}-E_{X_q}-1)-1)=\Phi_T(\Phi_T(-\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1))-1)=\Phi_T(-\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1))$ [cf. Proposition 1]

and therefore

$S_1=\Phi_T(\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)+\Phi_T(-\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)))$;

three cases may be distinguished as follows:

— if $\widehat{M}_{X_q}>\widehat{M}_{X_i}$, then $\widehat{M}_{X_q}-\widehat{M}_{X_i}-1\geq 0$ and $\widehat{M}_{X_i}-\widehat{M}_{X_q}-1\leq -2$ (as $\widehat{M}_{X_q}$ and $\widehat{M}_{X_i}$ are integers) $\Longrightarrow \Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)=1$, $\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)=0$ and $\Phi_T(-\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1))=\Phi_T(-1)=0$ [cf. function (2)],
and consequently
$S_1=\Phi_T(1)=1$

— if $\widehat{M}_{X_q}>\widehat{M}_{X_i}$, then $\widehat{M}_{X_q}-\widehat{M}_{X_i}-1\leq -2$ and $\widehat{M}_{X_i}-\widehat{M}_{X_q}-1\geq 0$(as $\widehat{M}_{X_q}$ and $\widehat{M}_{X_i}$ are integers) $\Longrightarrow \Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)=0$, $\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)=1$ and $\Phi_T(-\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1))=\Phi_T(-1)=0$ [cf. function (2)],
and consequently
$S_1=\Phi_T(-1)=0$

— if $\widehat{M}_{X_q}=\widehat{M}_{X_i}$, then $\widehat{M}_{X_q}-\widehat{M}_{X_i}-1=\widehat{M}_{X_i}-\widehat{M}_{X_q}-1=-1\Longrightarrow \Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)=\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)=\Phi_T(-1)=0$ and $\Phi_T(-\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1))=\Phi_T(0)=1$ [cf. function (2)],
and consequently
$S_1=\Phi_T(1)=1$.

To summarize

$$S_1=\begin{cases}1, & \text{if } (E_{X_q}>E_{X_i}) \\ & \quad \text{or } (E_{X_q}=E_{X_i} \text{ and } \widehat{M}_{X_q}\geq\widehat{M}_{X_i}) \\ 0, & \text{if } (E_{X_q}=E_{X_i}) \\ & \quad \text{or } (E_{X_q}=E_{X_i} \text{ and } \widehat{M}_{X_q}<\widehat{M}_{X_i}).\end{cases}$$

Taking into account Definition 4 and Proposition 2, we have

$$S_1=\begin{cases}1, & \text{if } |X_q|\geq|X_i| \\ 0, & \text{if } |X_q|<|X_i|\end{cases}$$

According to Definition 1, we may write

$$\operatorname{comp}(|X_i|,|X_q|)=S_1 \quad \text{if } i<q$$

• For $i>q$

$E_{x_q}>E_{x_i}\Longrightarrow E_{x_q}-E_{x_i}\geq 1$ and $E_{x_i}-E_{x_q}\leq -1$ (as E_{x_q} and E_{x_i} are integers) $\Longrightarrow \Phi_T(E_{X_q}-E_{X_i}-1)=1$ and $\Phi_T(E_{X_i}-E_{X_q}-1)=0$ (cf. function (2)),
which leads to:

$\Phi_T(-\Phi_T(E_{X_q}-E_{X_i}-1)-\Phi_T(E_{X_i}-E_{X_q}-1)+\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-1)=\Phi_T(\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-2)=0$ (as $\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-2\leq -1$), $\Phi_T(-\Phi_T(E_{X_q}-E_{X_i}-1)-\Phi_T(E_{X_i}-E_{X_q}-1)+\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-1)=\Phi_T(\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-2)=0$ (as $\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-2\leq -1$))
and consequently:
$S_2=\Phi_T(1-1)=\Phi_T(0)=1$ (cf. function (2))

$E_{x_q}>E_{x_i}\Longrightarrow E_{x_i}-E_{x_q}\geq 1$ and $E_{x_q}-E_{x_i}\leq -1$ (as E_{x_q} and E_{x_i} are integers) $\Longrightarrow \Phi_T(E_{X_q}-E_{X_i}-1)=0$ and $\Phi_T(E_{X_i}-E_{X_q}-1)=1$ (cf. function (2)),
which leads to:

$\Phi_T(-\Phi_T(E_{X_q}-E_{X_i}-1)-\Phi_T(E_{X_i}-E_{X_q}-1)+\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-1)=\Phi_T(\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-2)=0$ (as $\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-2\leq -1$), $\Phi_T(-\Phi_T(E_{X_q}-E_{X_i}-1)-\Phi_T(E_{X_i}-E_{X_q}-1)+\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-1)=\Phi_T(\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-2)=0$ (as $\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-2\leq -1$))
and consequently
$S_2=\Phi_T(-1-1)=\Phi_T(-2)=0$ (cf. function (2))

$E_{x_q}=E_{x_i}\Longrightarrow E_{x_i}-E_{x_q}-1=E_{X_q}-E_{X_i}-1=-1\Longrightarrow \Phi_T(E_{x_q}-E_{x_i}-1)=\Phi_T(E_{X_i}-E_{X_q}-1)=\Phi_T(-1)=0$ (cf. function (2)),
which leads to:

$\Phi_T(-\Phi_T(E_{X_q}-E_{X_i}-1)-\Phi_T(E_{X_i}-E_{X_q}-1)+\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-1)=\Phi_T(\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-1)=\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)$ (cf. Proposition 1), $\Phi_T(-\Phi_T(E_{X_q}-E_{X_i}-1)-\Phi_T E_{X_i}-E_{X_q}-1)+\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-1)=\Phi_T(\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-1)=\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)$ (cf. Proposition 1)
and therefore:
$S_2=\Phi_T(\Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)-\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)-1)$.

Three cases may be distinguished:

— If $\widehat{M}_{X_q}>\widehat{M}_{X_i}$, then $\widehat{M}_{X_q}-\widehat{M}_{X_i}-1\geq 0$ and $\widehat{M}_{X_i}-\widehat{M}_{X_q}-1\leq -2$ (as $\widehat{M}_{X_q}$ and $\widehat{M}_{X_i}$ are integers) $\Longrightarrow \Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)=1$, and $\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)=0$
and consequently
$S_2=\Phi_T(1-1)=\Phi_T(0)=1$

— If $\widehat{M}_{X_q}<\widehat{M}_{X_i}$, then $\widehat{M}_{X_q}-\widehat{M}_{X_i}-1\leq -2$ and $\widehat{M}_{X_i}-\widehat{M}_{X_q}-1\geq 0$ (as $\widehat{M}_{X_q}$ and $\widehat{M}_{X_i}$ are integers) $\Longrightarrow \Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)=0$, $\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)=1$
and consequently
$S_2=\Phi_T(-1-1)=\Phi_T(-2)=0$

— If $\widehat{M}_{X_q}=\widehat{M}_{X_i}$, then $\widehat{M}_{X_q}-\widehat{M}_{X_i}-1=\widehat{M}_{X_i}-\widehat{M}_{X_q}-1=-1\Longrightarrow \Phi_T(\widehat{M}_{X_q}-\widehat{M}_{X_i}-1)=\Phi_T(\widehat{M}_{X_i}-\widehat{M}_{X_q}-1)=\Phi_T(-1)=0$

and consequently
$S_2 = \Phi_T(-1) = 0$
To summarize

$$S_2 = \begin{cases} 1, & \text{if } (E_{X_q} > E_{X_i}) \\ & \quad \text{or } (E_{X_q} = E_{X_i} \text{ and } \widehat{M}_{X_q} > \widehat{M}_{X_i}) \\ 0, & \text{if } (E_{X_q} < E_{X_i}) \\ & \quad \text{or} (E_{X_q} = E_{X_i} \text{ and } \widehat{M}_{X_q} \leq \widehat{M}_{X_i}) \end{cases}.$$

Taking into account Definition 4 and Proposition 2, we have

$$S_2 = \begin{cases} 1, & \text{if } |X_q| > |X_i| \\ 0, & \text{if } |X_q| \leq |X_i|. \end{cases}$$

According to Definition 1, we may write

$$\text{comp}\,(|X_i|, |X_q|) = S_2 \quad \text{if } i < q.$$

■

APPENDIX II

PROOF OF PROPOSITION 4

Four cases can be distinguished.

1) $S_{X_q} = S_{X_i} = 0 \Longrightarrow \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}(|X_i|, |X_q|) - 1) = \Phi_T(\text{comp}(|X_i|, |X_q|) - 1) = \text{comp}(|X_i|, |X_q|)$ (cf. Proposition 1), $\Phi_T(S_{X_q} + S_{X_i} - \text{comp}(|X_i|, |X_q|) - 2) = \Phi_T(-\text{comp}(|X_i|, |X_q|) - 2) = 0$ (as $-\text{comp}(|X_i|, |X_q|) - 2 \leq -2$), and $\Phi_T(-S_{X_q} + S_{X_i} - 1) = \Phi_T(-1) = 0$
and consequently
$\Phi_T(-S_{X_q} - S_{X_i} + \text{comp}(|X_i|, |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}(|X_i|, |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = \text{comp}(|X_i|, |X_q|)$.
As X_q and X_i are simultaneously positive, i.e., $|X_q| = X_q$ and $|X_i| = X_i$, we have $\Phi_T(-S_{X_q} - S_{X_i} + \text{comp}(|X_i|, |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}(|X_i|, |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = \text{comp}(X_i, X_q)$.
2) $S_{X_q} = S_{X_i} = 1 \Longrightarrow \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}(|X_i|, |X_q|) - 1) = \Phi_T(\text{comp}(|X_i|, |X_q|) - 3) = 0$ (as $\text{comp}(|X_i|, |X_q|) - 3 \leq -2$), $\Phi_T(S_{X_q} + S_{X_i} - \text{comp}(|X_i|, |X_q|) - 2) = \Phi_T(-\text{comp}(|X_i|, |X_q|))$ and $\Phi_T(-S_{X_q} + S_{X_i} - 1) = \Phi_T(-1) = 0$, and consequently (see equation at the bottom of the page).

For $i < q$

$$\begin{aligned} 1 - \text{comp}\,(|X_i|, |X_q|) &= \begin{cases} 1, & \text{if } |X_i| < |X_q| \\ 0, & \text{if } |X_i| \geq |X_q| \end{cases} \\ &= \begin{cases} 1, & \text{if } |X_i| > |X_q| \\ 0, & \text{if } |X_i| \leq |X_q| \end{cases} \\ &= \text{comp}(X_i, X_q) \\ &\qquad \text{(cf. Definition 1 and (15))} \end{aligned}$$

and for $i > q$

$$\begin{aligned} 1 - \text{comp}\,(|X_i|, |X_q|) &= \begin{cases} 1 & \text{if } |X_i| \leq |X_q| \\ 0 & \text{if } |X_i| > |X_q| \end{cases} \\ &= \begin{cases} 1 & \text{if } |X_i| \geq |X_q| \\ 0 & \text{if } |X_i| < |X_q| \end{cases} \\ &= \text{comp}(X_i, X_q) \\ &\qquad \text{(cf. Definition 1 and (16))} \end{aligned}$$

which results in $\Phi_T(-S_{X_q} - S_{X_i} + \text{comp}(|X_i|, |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}(|X_i|, |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = \text{comp}(X_i, X_q)$.
3) $S_{X_q} = 1$ and $S_{X_i} = 0$ (i.e., $X_q < X_i$ (cf. Definition 5)) $\Longrightarrow \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}(|X_i|, |X_q|) - 1) = \Phi_T(\text{comp}(|X_i|, |X_q|) - 2) = 0$ (as $\text{comp}(|X_i|, |X_q|) - 2 \leq -1$)), $\Phi_T(S_{X_q} + S_{X_i} - \text{comp}(|X_i|, |X_q|) - 2) = \Phi_T(-\text{comp}(|X_i|, |X_q|) - 1) = 0$ (as $-\text{comp}(|X_i|, |X_q|) - 1 \leq -1$) and $\Phi_T(-S_{X_q} + S_{X_i} - 1) = \Phi_T(-2) = 0$ and consequently
$\Phi_T(-S_{X_q} - S_{X_i} + \text{comp}(|X_i|, |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}(|X_i|, |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = 0 = \text{comp}(X_i, X_q)$ (cf. Definitions 1 and 5).
4) $S_{X_q} = 0$ and $S_{X_i} = 1$ (i.e., $X_q > X_i$ (cf. Definition 5)) $\Longrightarrow \Phi_T(-S_{X_q} - S_{X_i} + \text{comp}(|X_i|, |X_q|) - 1) = \Phi_T(\text{comp}(|X_i|, |X_q|) - 2) = 0$ (as $\text{comp}(|X_i|, |X_q|) - 2 \leq -1$), $\Phi_T(S_{X_q} + S_{X_i} - \text{comp}(|X_i|, |X_q|) - 2) = \Phi_T(-\text{comp}(|X_i|, |X_q|) - 1) = 0$ (as $-\text{comp}(|X_i|, |X_q|) - 1 \leq -1$)) and $\Phi_T(-S_{X_q} + S_{X_i} - 1) = \Phi_T(0) = 1$ and consequently
$\Phi_T(-S_{X_q} - S_{X_i} + \text{comp}(|X_i|, |X_q|) - 1) + \Phi_T(S_{X_q} + S_{X_i} - \text{comp}(|X_i|, |X_q|) - 2) + \Phi_T(-S_{X_q} + S_{X_i} - 1) = 1 = \text{comp}(X_i, X_q)$ (cf. Definitions 1 and 5).

■

$$\begin{aligned} \Phi_T\left(-S_{X_q} - S_{X_i} + \text{comp}\,(|X_i|, |X_q|) - 1\right) + & \\ \Phi_T\left(S_{X_q} + S_{X_i} - \text{comp}\,(|X_i|, |X_q|) - 2\right) + & \\ \Phi_T\left(-S_{X_q} + S_{X_i} - 1\right) &= \Phi_T\left(-\text{comp}\,(|X_i|, |X_q|)\right) \\ &= \begin{cases} 1 & \text{if comp}\,(|X_i|, |X_q|) = 0 \\ 0 & \text{if comp}\,(|X_i|, |X_q|) = 1 \end{cases} \\ &= 1 - \text{comp}\,(|X_i|, |X_q|) \end{aligned}$$

REFERENCES

[1] Y. S. Abu-Mostafa and D. Pslatis, "Optical neural computers," *Sci. Amer.*, vol. 256, pp. 88–95, Mar. 1987.

[2] F. Ali and T. Pavlidis, "Syntactic recognition of handwritten numerals," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 537–541, July 1977.

[3] J. A. Anderson, "A simple neural network generating an interactive memory," *Math. Biosci.*, vol. 14, pp. 197–220, 1972.

[4] E. Ataman, V. K. Aatre, and K. M. Wong, "A fast method for real-time median filtering," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-28, pp. 415–421, Aug. 1980.

[5] T. E. Bell, "Optical computing: A field in flux," *IEEE Spectrum*, vol. 23, pp. 34–57, Aug. 1986.

[6] Y. Ben-Asher, D. Peleg, R. Ramaswami, and A. Schuster, "The power of reconfiguration," *J. Parallel Distrib. Comput.*, vol. 13, no. 2, pp. 139–153, 1991.

[7] Y. Ben-Asher and A. Schuster, "Optical splitting graphs," presented at the Int. Topical Meeting Optical Computing, Kobe, Japan, 1990.

[8] L. N. Binh and H. C. Chong, "A neural-network contention controller for packet switching networks," *IEEE Trans. Neural Networks*, vol. 6, pp. 1402–1410, Nov. 1995.

[9] A. C. Bovik, T. S. Huang, and D. C. Munson, "A generalization of median filtering using linear combinations of order statistics," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-31, pp. 1342–1350, 1983.

[10] T. X. Brown, "Neural networks for switching," *IEEE Commun. Mag.*, vol. 27, pp. 72–81, Nov. 1989.

[11] T. X. Brown and K. H. Liu, "Neural-network design of a Banyan network controller," *IEEE J. Select. Areas Commun.*, vol. 8, pp. 1428–1473, 1990.

[12] T. X. Brown, "Neural-network design for switching network control," Ph.D. dissertation, California Inst. Technol., Pasadena, CA, 1990.

[13] ——, "Neural networks for switching," in *Neural Networks in Telecommunications*, B. Yuhas and N. Ansari, Eds. Boston, MA: Kluwer, 1994.

[14] B. D. Calvert and C. A. Marinov, "Another K-winner-take-all analog neural network," *IEEE Trans. Neural Networks*, vol. 11, pp. 829–838, July 2000.

[15] A. K. Chandra, L. Stockmeyer, and U. Vishkin, "Constant depth reducibility," *SIAM J. Comput.*, vol. 13, pp. 423–439, 1984.

[16] W. T. Chen, P. Gader, and H. Shi, "Lexicon-driven handwritten word recognition using optimal linear combinations of order statistics," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, pp. 77–82, Jan. 1999.

[17] A. Cichocki and R. Unbehauen, "Neural networks for solving systems of linear equations and related problems," *IEEE Trans. Circuits Syst.*, vol. 39, pp. 124–138, Feb. 1992.

[18] J. Choi and B. J. Shen, "A high-precision VLSI winner-take-all circuit for self-organizing neural networks," *IEEE J. Solid-State Circuits*, vol. 28, pp. 576–583, May 1993.

[19] H. Elgindy and P. Wegrowicz, "Selection on the reconfigurable mesh," in *Proc. Int. Conf. Parallel Processing*, Aug. 1991, pp. III.26–III.33.

[20] R. Erlanson and Y. Abu-Mustapha, "Analog neural networks as decoders," in *Advances in Neural Information Processing System*, D. S. Touretzky, Ed. Los Altos, CA: Morgan Kaufmann, 1991, vol. 3, pp. 585–588.

[21] D. S. Fukushima, "A neural network for visual pattern recognition," *IEEE Comput.*, vol. 21, pp. 65–75, Mar. 1988.

[22] K. Fukushima and N. Wake, "Handwritten alphanumeric character recognition by the neocognition," *IEEE Trans. Neural Networks*, vol. 2, pp. 355–365, May 1991.

[23] M. Furst, J. B. Saxe, and M. Sipser, "Parity, circuits and the polynomial-time hierarchy," in *Proc. 22nd IEEE Symp. Foundations Computer Science*, 1981, pp. 260–270.

[24] E. Hao, P. D. MacKenzie, and Q. F. Stout, "Selection on the reconfigurable mesh," in *Proc. Frontiers Massively Parallel Computation*, Oct. 1992, pp. 38–45.

[25] J. J. Hopfield and D. W. Tank, "Simple 'Neural' optimization networks an A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 533–541, May 1986.

[26] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational ability," in *Proc. Nat. Academy Science USA*, vol. 79, Apr. 1982, pp. 2554–2558.

[27] T. S. Huang, *Two-Dimensional Digital Signal Processing II: Transforms and Median Filters*. New York: Springer-Verlag, 1981.

[28] T. S. Huang, G. J. Yang, and G. Y. Tang, "A fast two-dimensional median filtering algorithm," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-27, pp. 13–18, Feb. 1979.

[29] L. G. Johnson and S. M. S. Jalaleddine, "MOS implementation of winner-take-all network with application to content-adressable memory," *Electron. Lett.*, vol. 27, no. 11, pp. 957–958, May 1991.

[30] J. Jang and V. K. Prasana, "An optimal sorting algorithm on reconfigurable mesh," *J. Parallel and Distributed Computing*, vol. 25, pp. 31–41, Feb 1995.

[31] M. P. Kenedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 554–562, May 1988.

[32] T. Kohonen, "Correlation matrix memories," *IEEE Trans. Comput.*, vol. C-21, pp. 353–359, 1972.

[33] H. K. Kwan, "One-layer feedforward neural network fast maximum/minimum determination," *Electron. Lett.*, pp. 1583–1585, 1992.

[34] J. Lazaro, S. Ryckebush, M. A. Mahowald, and C. A. Mead, "Winner-take-all networks of $O(N)$ complexity," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed. Los Altos, CA: Morgan Kaufmann, 1989, vol. 1, pp. 703–711.

[35] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Handwritten zip code recognition with multilayer networks," in *Proc. 10th Int. Conf. Pattern Recognition*, 1990, pp. 35–40.

[36] S. Lee and J. F. Pan, "Offline tracing and representation of signatures," *IEEE Trans. Syst., Man., Cybern.*, vol. 22, pp. 755–771, July/Aug. 1992.

[37] Y. Lee, "Handwritten digit recognition using K nearest-neighbor, radial-basisfunction, and back-propagation networks," *Neural Comput.*, vol. 3, pp. 440–449, 1991.

[38] Y. H. Lee and A. T. Fam, "An edge gradient enhancing adaptive order statistic filter," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-35, pp. 680–695, 1987.

[39] S. Lee and J. C. J. Pan, "Unconstrained handwritten numeral recognition based on radial basis competitive and cooperative networks with spatio-temporal feature representation," *IEEE Trans. Neural Networks*, vol. 7, Mar. 1996.

[40] C. H. Leung, Y. S. Cheung, and Y. L. Wong, "A knowledge-based stroke-matching method for Chinese character recognition," *IEEE Trans. Syst., Man., Cybern.*, vol. SMC-11, pp. 993–1003, Nov./Dec. 1987.

[41] J. Levinson, I. Kuroda, and T. Nishitani, "A reconfigurable processor array with routing LSI's and general purpose DSPs," in *Proc. Int. Conf. Application Specific Array Processors*, Oct. 1992.

[42] H. Li and M. Maresca, "Polymorphic-torus network," *IEEE Trans. Comput.*, vol. 38, pp. 1345–1351, Sept. 1989.

[43] R. Lin, S. Olariu, J. Schwing, and J. Zhang, "A VLSI-optimal constant time sorting on reconfigurable mesh," in *Proc. 9th Eur. Workshop Parallel Computing*, Spain, 1992, pp. 1–16.

[44] R. P. Lippman, "An introduction to computing with neural nets," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 35, pp. 2–44, Apr. 1987.

[45] G. L. Martin and J. A. Pittman, "Recognizing hand-printed letters and digits using backpropagation learning," *Neural Comput.*, vol. 3, pp. 258–267, 1991.

[46] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[47] C. Mead and M. Ismail, *Analog VLSI Implementation of Neural Systems*. Norwell, MA: Kluwer, 1989.

[48] D. Del Corso, K. E. Grosspietsh, and P. Treleaveng, "Silicon neural networks," *Special Issue on a Collection of Good Papers on Digital and Analog Artificial Neural Networks, IEEE Micro*, vol. 9, Dec. 1989.

[49] M. Mestari and A. Namir, "AMAXNET: A neural network implementation of adjustable MAXNET in fixed time," in *Proc. IFAC-IFIP-IMACS Int. Conf. Control Industrial Systems*, vol. 2, Belfort, France, May 20–22, 1997, pp. 543–549.

[50] ——, "MinMaxNet: A neural network implementation of min/max filters," in *IFIP Proc. Int. Conf. Optimization-Based Computer-Aided Modeling Design*, vol. 1, Noisy-le-Grand, Paris, France, May 28–30, 1996, pp. 26.1–26.4.

[51] ——, "AOSNET: A neural network implementation of adjustable order statistic filters in fixed time," *SIAMS J.*, vol. 36, pp. 509–535, 2000.

[52] ——, "A neural network implementation of L_∞, metric partitional clustering in fixed time," *SIAMS J.*, vol. 41, no. 2, pp. 351–380, 2001.

[53] M. Mestari, A. Namir, and J. Abouir, "Switched capacitor neural networks for optimal control of nonlinear dynamic systems: Design and stability analysis," *SIAMS J.*, vol. 41, no. 3, pp. 559–591, 2001.

[54] R. Miler, V. K. Prasana Kumar, D. I. Reisis, and Q. F. Stout, "Meshes with reconfigurable buses," in *Proc. MIT Conf. Advanced Research VLSI*, Apr. 1988, pp. 163–178.

[55] Y. Nakagawa and A. Rosenfeld, "A note of the use of local min and max operations in digital picture processing," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 632–635, 1978.

[56] K. Nakano, T. Msuzawa, and N. Tokura, "A sub-logarithmic time sorting algorithm on a reconfigurable array," *IEICE*, vol. E-74, no. 11, pp. 3 894–3 901, Nov. 1991.
[57] Y. Nakatani, D. Sasaki, Y. Liguni, and H. Maeda, "Online recognition of handwritten Hiragana characters based upon a complex autoregressive model," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 21, no. 1, pp. 73–76, Jan. 1999.
[58] P. M. Narendra, "A seperable median filter for image noise smoothing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, pp. 20–29, Jan. 1981.
[59] M. Nigam and S. Sahni, "Sorting n numbers on $n \times n$ reconfigurable meshes with buses," in *Proc. Int. Parallel Processing Symp.*, Apr. 1993, pp. 174–181.
[60] M. Parizeau and R. Plamondon, "A comparative analysis of regional correlation, dynamic time warping and skeletal tree matching for signature verification," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 710–717, July 1990.
[61] O. Y. Pecht and M. Gur, "A biologically-inspired improved MAXNET," *IEEE Trans. Neural Networks*, vol. 6, pp. 757–759, May 1995.
[62] J. Pankove, C. Radehaus, and K. Wanger, "Winner-take-all neural net with memory," *Electron. Lett.*, vol. 26, no. 6, pp. 349–350, Mar. 1990.
[63] V. K. Prasana Kumar and C. S. Ragavendra, "Array processor with multiple broadcasting," *J. Parallel Distrib. Comput.*, vol. 4, pp. 173–190, 1987.
[64] A. Rajavelu, M. T. Musavi, and M. V. Shirvaikar, "A neural-network approach to character recognition," *Neural Networks*, vol. 2, pp. 387–393, 1989.
[65] D. S. Richrad, "VLSI median filters," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 38, pp. 145–153, Jan. 1990.
[66] J. Rothsten, "Bus automata, brains, and mental model," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 522–531, Apr. 1988.
[67] G. Seiler and J. A. Nossek, "Winner-take-all cellular neural networks," *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 184–194, Mar. 1993.
[68] P. Shi and R. K. Ward, "A neural network implementation of median filtering," in *IEEE Pacific Rim Conf.*, Victoria, BC, Canada, 1989.
[69] L. Snyder, "Introduction to the reconfigurable highly parallel computer," *Comput.*, vol. 15, no. 1, pp. 47–56, Jan. 1982.
[70] Y. Suganuma, "Learning structures of visual patterns from single instances," *Artif. Intell.*, vol. 50, pp. l–36, 1991.
[71] B. W. Suter and M. Kabrisky, "On a magnitude preserving iterative Maxnet algorithm," *Neural Comput.*, vol. 4, pp. 224–233, 1992.
[72] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in on-line handwriting recognition," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 787–808, Aug. 1990.
[73] C. Toumazou, F. J. Lidgey, and D. G. Haigh, *Analog IC Design: The Current-Mode Approach*. Stevenage, U.K.: Perigrinus, 1990.
[74] K. Urahama and T. Nagao, "K-winners-take-all circuit with $O(N)$ complexity," *IEEE Trans. Neural Networks*, vol. 6, pp. 776–778, May 1995.
[75] T. Wakahara, H. Murase, and K. Odaka, "On-line handwriting recognition," *Proc. IEEE*, pp. 1181–1194, July 1992.
[76] B. F. Wang, G. H. Chen, and F. C. Lin, "Constant time sorting on a processor array with a reconfigurable bus systems," *Inform. Processing Lett.*, pp. 187–192, 1990.
[77] C. C. Weems and J. H. Burill, "The image understanding architecture and its programming environment," in *Parallel Architectures and Algorithms for Image Understanding*, V. K. Prasana Kumar, Ed. New York: Academic, 1991.
[78] W. E. Weideman, M. T. Manry, H. C. Yau, and W. Gong, "Comparisons of a neural network and a nearest-neighbor classifier via the numeric handprint recognition problem," *IEEE Trans. Neural Networks*, vol. 6, pp. 1524–1530, Nov. 1995.
[79] J. H. Winters and C. Rose, "Minimum distance automata in parallel networks for optimum classification," *Neural Networks*, vol. 2, pp. 127–132, 1989.
[80] J. Wook, M. Nigam, V. K. Prasana, and S. Sahni, "Constant time algorithms for computational geometry on the reconfigurable mesh," *IEEE Trans. Parallel Distrib. Syst.*, vol. 8, pp. 1–12, Jan. 1997.
[81] J. F. Yang and C. M. Chen, "A general mean based iteration winner-take-all neural network," *IEEE Trans. Neural Networks*, vol. 6, pp. 14–24, Jan. 1995.
[82] J. F. Yang, C. M. Chen, W. C. Wang, and J. Y. Lee, "An improved general mean based iteration winner-take-all neural network," in *Proc. Int. Symp. Artificial Neural Network*, 1994, pp. 429–434.
[83] J. F. Yang and C. M. Chen, "Winner-take-all neural network using the highest threshold," *IEEE Trans. Neural Networks*, vol. 11, pp. 194–199, Jan. 2000.
[84] J. C. Yen and S. Chang, "Improved winner-take-all neural network," *Electron. Lett.*, pp. 662–664, Mar. 1992.
[85] J. C. Yen, F. J. Chang, and S. Chang, "A new winner-take-all architecture in artificial neural networks," *IEEE Trans. Neural Networks*, vol. 5, pp. 838–843, Sept. 1994.
[86] S. Zunino, "Circuit implementation of the K-winner machine," *Electron. Lett.*, vol. 35, no. 14, pp. 1172–1173, July 8, 1999.

Mohammed Mestari received the M.A. degree from ENSET, Mohammedia, Morocco, in 1991 and the Ph.D. degree in applied mathematics and the Ph.D. degree in artificial intelligence from Hessan II University, Faculty of Science Ben M'Sik Casablanca, Casablanca, in 1997 and 2000, respectively.

He is currently an Assistant Professor of Applied Mathematics at ENSET, Mohammedia, and a Member of the Fundamental Research Unit (UFR) of mathematics applied to engineering sciences. His research interests include neural networks for signal processing, neural networks hardware implementation, high-speed and/or low-power techniques and systems for neural networks, and theoretical issues directly related to hardware implementation.