Applied Mathematical Sciences, Vol. 3, 2009, no. 29, 1407 - 1451

A Neural-Network to Solving the Output Contention in Packet Switching Networks

A. Badi and K. Akodadi

Département de Mathématiques et Informatique
Faculté des sciences Ben M'sik Université Hassan II, Mohammedia
B.P 7955, Sidi Othman, Casablanca, Morocco
abdelgh66@hotmail.com

M. Mestari

E.N.S.E.T Mohammedia Avenue Hssan II Mohammedia Morocco

A. Namir

Département de Mathématiques et Informatique
Faculté des sciences Ben M'sik Université Hassan II, Mohammedia
B.P 7955, Sidi Othman, Casablanca, Morocco
a.Namir@yahoo.fr

Abstract

Optical packet switching based on Wavelength Division Multiplexing (WDM) offers the possibility of increased granularity and more effective use bandwidth in large capacity systems, on scale of Tb/s. The natural integration of optical packet switching in photonic technology opens up the possibility of packet switching in transparent optical channels, where the packets remain from end-to-end in the optical domain, without the necessity of optoelectronic conversion. Therefore, the optical switching must be robust enough in order to provide conditions to solve the contention between optical packets in access networks, where the traffic is less aggregated. This works presents a novel approach to solving the output contention in optical packet switching networks with synchronous switching mode. A contention controller has been designed based on the Order Statics Filters (OSF) neural-network technique with a speed up factor to achieve a real-time computation of a non blocking switching high-speed high-capacity packet switch without packet loss. A neural network, OSF, which with any of binary as input, outputs

the K^{th} largest element of the array is proposed. Overall time is constant, and does not depend upon the size of the input array, being just eleven times the processing time for a single neuron. This neural network may be used as a building block for hardware implementation of order statistic filters.

Keywords: Optical packets switching (OPS), Head-of-line (HOL), Multiprotocol label switching (MPLS), Order static filter (OSF), Wavelength division multiplexing (WDM), generalized multi-protocol label switching (GMPLS), First in first out (FIFO), Pattern recognition and image processing (PRIP)

1 Introduction

Output contention is an inherent characteristic of packet switching networks due to unscheduled arrival process of information packets. In packet switch with synchronous switching mode, there may be up to N number of packets simultaneously contending for the same output port but only up to C contending packets (C is less than N) can be routed out to that output port where C is the speed up factor of the switch. The principal tasks of the contention controller are to select up to C contending packets for each output port and to switch selected packets to their destined output ports. Such tasks must be completed in less time than it takes to transmit a packet.

Neural networks have been proposed for solving the output-contention problem. Troudet et al. [71], Marrakchi et al. [38] have proposed the use of a Hopfield neural network for real time control of input queued crossbar switch for switching packets at maximum throuput. Brown et al. [10] has proposed a neural network based on multiple overlapping winner-take-all circuits to compute a nonblocking switching configuration for an input queued Banyan switch with a performance of within a factor of two of the nonblocking switch. Le Nguen Binh et al. [8] have proposed K-winner neural network [75] for solving the output contention in packet switching networks with switching mode. Unlike Hopfield energy functions approach or K-winer-take-all, that requires the researcher to first define the constraints of the problem and then go through an imprecise and obscuring energy function to define the weights, these networks have properties that can be directly defined and controlled. This direct approach allows efficient implementations that are scalable to large size and as long as the external inputs are within defined limits, the network will always satisfy the constraints embodied in the OSF (priority). Furthermore our proposed scheme demonstrates, through simulation, the behaviour of packet switching with the speed up factor C. A contention controller using the OSF

neural network technique is thus proposed and analysed.

In our proposal, the search for the contending C packets for size N is carried out in 11τ time and only requires employment of N sorting networks.

In this paper, we proposed the use of OSF is a technique extensively used in pattern recognition and image processing PRIP applications [2], [13], [10], [11], [33], [41]-[53], [55], [57], [63], [67], [70], [74]. During the past decades, considerable efforts have been devoted to developing special computer architectures for PRIP applications [5], [34], [35], [54], [61], [64], [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 sorting and adaptive OSF (called comparison and selection filters [33]). The main task of the OSF is to find the k^{th}-order statistic of an input array, defined as being the k^{th} largest element in the array. This technique finds application in telecommunications particularly for controlling data packet switches [8], [9]-[12]. In [32], a member of the OSF family shows applications in VLSI auditory and visual systems, while in [21], another filter of OSF family as an analogue decoder of err-correcting codes is proposed. OSF are mostly implemented in software [2], [4], [20], [24], [26], [28], [36], [56], [59], [61], [70], [73], [74], whereas hardware implementations are designed only for specific members, particularly the median and maximum filters, of the OSF family [17], [29], [41], [42], [44], [60], [66], [77].

All neural networks considered in this paper have a feed-for-ward structure with two kinds of neurons, linear and threshold 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 increase, 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 OSF, however, only grows quadratically.

Neural networks are noted for their ability to process large amounts of data quickly using a copious number of highly find solutions; the multitude of readily available data; and the similar neural and switching network topologies all suggest the potential for exploiting neural networks in the switching control problem. Finally, the significance and potential of the neural-network are discussed and concluded.

The rest of paper is organised as follows. In section II, we describe switching packet. Section III is devoted to the application of the OSF in implementation of various order statistic filter, including sorting and adaptive order statistic filters. Section IV developing a simulating system. Section V contains the conclusion.

2 Optical Packet Switching and Contention Resolution Problems

2.1 Background

This section will introduce some basic concepts of switching theory. More detailed development of switching the theory can be found in any of several references, [6], [27] and [65]. We abstract a switch as a device that takes a set of N inputs and reproduces them in any permuted order at the output. We restrict our discussion to square switches, where the number of inputs and outputs is the same, although the concepts that we develop readily generalize to non-square switches. If the inputs and outputs of the switch are N distinct lines, then we refer to the switch as a space switch. Alternatively the switch could one line for the input one line for the output. In this case, the inputs and outputs are N distinct blocks of data in witch the order that the block are sent is permuted by the switch. Since each time switch is equivalent to some space switch, we will restrict our discussion to space switches.

The basic switch is the NxN crossbar switch. Conceptually, it comprises a packet request in a general switch is a request for a connection from one unused input to one unused output. A packet is blocked if the connection can not be put up through the switch. This occurs due to constraints from the architecture of the switch, or due to the current state of the switch. A switch is non-blocking if simultaneously given any legal set of packets (each input and output used at most once); the switch can put all of the packets. A switch is strictly non-blocking in any sequence of legal packet requests, with

intervening packet disconnects, can be put up as each request arrives, and no matter what algorithm is used for routing packets. The NxN crossbar is the prototypical example of the non-blocking class. While such a switch is desirable, unfortunately it is at the expense of N^2 crospoints. The crospoint count of switch is often used as measure of its cost or complexity. We desire to reduce the number of crospoints. Usually this is achieved by building larger switches stages of smaller crossbar switches.

2.2 Contention resolution problems

WDM has been rapidly acceptance as the technology that is able to handle the forecast dramatic increase of bandwidth demand in future networks [30]. Besides the huge amounts of bandwith, all-optical WDM networks also allow high-speed data transmission without electronic converts at intermediate nodes and transparency with respect to data format to be achieved [62]. However, the service evolution and the rapid increase in traffic levels fuel the interest on optical packet switching resolution problem. While current applications of WDM focus on the fast static usage of individual WDM channels, optical packet switching technologies enable the fast allocation of WDM channels in an on-demand fashion with fine granularities. The challenge now is to combine the advantages of the relatively coarse-grained WDM techniques with emerging with optical switching capabilities to yield a high-throughput optical platform.

A further reason leading to optical packet switching is its intrinsic flexibility to cheaply support incremental increase of the transmission bit rate [68]. One of the key problems in application of packet switching in optical domain is the handling of packet contention that take place when two or more incoming packets are directed to the same output line. Contention can be solved in the following three ways; 1) by dropping the contending packet; 2) by deflecting the contending packet to another part of wavelength; 3) by buffering the contending packet (using delay lines or using electronic memory) until the output port is free. Of these three, only buffering offers an immediate solution as 1) port deflection is undesirable because it is in general not compatible with end-to end requirements as implemented in Generalized Multi-Protocol Label Switching (GMPLS) [7]; 2) by dropping the contending packet is obviously the poorest strategy, as it relies on the higher level protocols in retransmitting issues. Thus, buffering does not have any of the above mentioned drawbacks and moreover enables the prioritization of traffic based on, e.g., the differentiated services (DiffServ) specification [58]. The drawback of buffering is the introduction of additional delay and jitter in the network.

2.3 A novel approach to solving the output contention in packet switching networks

Output contention occurs when there are more than one (up to N) HOL packets from the input ports simultaneously requesting to be switched to the same output port. Only C of these contending packets can be routed to the same output port. The remaining of the contending packets have to wait for the next transmission cycle to compete again. The approach taken in resolving the output-contention problem and selecting up to C packets for each output port depends on the service selection policy employed by the packet switch. Typical service selections policies are the random selection policy, FIFO selection policy, longest queue selection policy, and customer priority selection policy. In packet transmission and switching, the sequence of the packets as transmitted by the customers must be maintained. This means that only the HOL packets in all the input buffers can compete in the selection process. The random selection policy stipulates that up to C packets can be routed to the same output port on a random basis with each of the contending packets having the equal probability of being selected. FIFO selection policy, on the other hand, dictates that packets which appear at the HOL positions at a given time slot that are destined for the same output are transferred before all the packets which appear at the HOL positions at the subsequent time slots with the same output destination. If there are several packets with the same arrival time, random selection is made between these packets. In the longest queue selection policy, the packet from the longest input queue is sent first. If there are several queues with the same maximum length, random selection is made between these contending packets. Under the customer priority selection policy, the higher priority packets are serviced ahead of the lower priority packets. This service selection policy is particularly important with regard to services that have a stringent delay requirement.

Our novel approach to resolving the output contention and implementing the service-selection policy is best described by to referring to figure. 1. Which shows how a request for packet transmission through an N x N crossbar can be mapped onto an N x N input request matrix. Each input and output connection of the nonblocking optoelectronic WDM or Time Division Multiplexing(TDM).

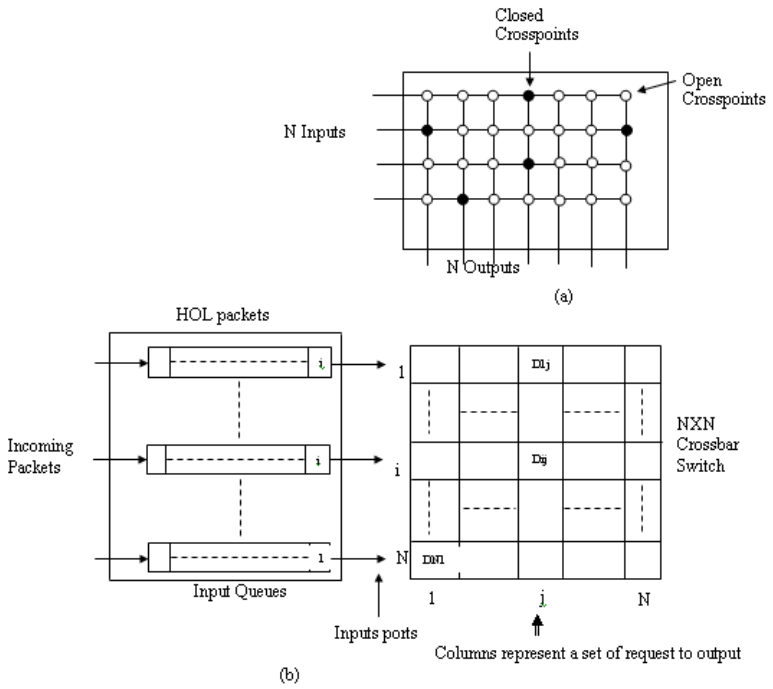


Figure 1: (a) A crossbar switch used to illustrate the input-output connections in the WDM packet switch. (b) Input request matrix describing the state of the packets in the HOL positions.

Packet switches can be represented by an imaginary crosspoint. These switches are important in multiwavelength WDM optical communication networks where the number of multiplexed optical wavelength channels, considered as wavelength packets, can be of order of a few hundred in the two optical transmission windows at 1300 and 1550 manometer regions. Therefore, a crossbar structure is used to facilitate our illustration of the input-output connections in a WDM packet switch. The N x N input request matrix describes the input request state of the HOL packets. The rows and columns of the input request matrix correspond to the input and output ports, respectively. If there is packet from input port i destined to output port j, then cell D_{ij} in the input request matrix is assigned a positive number, otherwise $D_{ij} = 0$. The value of D_{ij} will depends on the type of service selection policies used. A simple approach to resolve the output contention is to select up to C of the $D_{ij} > 0$ cells from each the column. If we were to stipulate that the task of selecting the successful packets for each output port (up to C packets) is to select the C largest D_{ij} values within each column, the value of D_{ij} is computed as follows. In a random service selection policy, the value of D_{ij} is a function of a random number. If we let U be the maximum positive value that

D_{ij} can take, then under the random service selection policy $Dij = r$, where r is a random number in the range $0 < r < U$, the value of D_{ij} is a function (denoted by f) of two variables

D_{ij}(FIFO) = f(waiting time, random number r)
D_{ij}(longest queue) = f(queue length, random number r)
D_{ij}(packet priority) = f(priority level, random number r).

The random number r is included to take into account the possibility of packets having the same waiting time, queue length, or priority level. The maximum contribution of the random number to the value of Dij must not exceed the minimum contribution from the waiting time, queue length, or packet priority. Therefore, the matrix cell value is given by

$$D_{ij} = P_s.Mult + r$$

Where P_s is the (HOL) packet's parameter as determined by the service discipline, Mult is a multiplier, and r is a random number. The random number, r is bounded by $0 < r < Mult$. The matrix cell value, D_{ij}, is bounded by $0 < D_{ij} \leq U$. The maximum value of r, i.e., Mult must be made as large as possible to reduce the probability of having packets with the same value of P_s and in the same column from having the same value of r. There is, however, a constraint placed on the value Mult, i.e., Mult $\leq$(maximum value of P_s). If such a possibility occurs, a random selection is made among them. For each column, the packet with the largest D_{ij} value is the first to be transmitted to their destined output ports.

The computation of this nonblocking connection configuration must be completed in less time than it takes to transmit a packet. The length of time allowed is determined by the packet length and the transmission bit rate, T = L/B where T is the packet duration, L is the packet length and B is the transmission bit rate. With an Asynchronous Transfer Mode (ATM) cell of 53 bytes and a bit rate of Gigabits/s, the packet duration is about 424 ns.

3 The OSF neural network for solving the contention problem

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

3.1 Neurons Used

OSF employs two kinds of neurons, both of which are commonly used in neural network applications [22], [31], [37], [41]-[53]. The only difference network between the two is in their activation function: one employs linear activation function and the other threshold-logic activation function. Their schematic representations are shown in Fig. 2(b), where y is the output. These two kinds of neurons sum the n weighted inputs and pass the result through a nonlinearity according to

$$y = \phi(\sum_{n}^{i=1} \omega_i x_i - \theta)$$

Where θ 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,...n), n is the number of inputs, and y represents the output [cf. Fig. 2(a)].

The threshold-logic neuron model [see Fig. 2(b)] uses only the binary (hard limiting) function [see Fig. 2(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 the equation

$$\phi(x) = \begin{cases} 1, if x \geq 0 \\ 0, otherwise \end{cases}$$

Where $x = \sum_{i=1}^{n} \omega_i x_i - \theta x$, and is the threshold-logic activation function or binary activation function [cf. Fig. 2(c)].

In the case of the linear activation function [see Fig. 2(b)], the output y is given by $\phi_L(x) = x$ Where ϕ_L is the linear activation function [see Fig. 2(c)] defined by $\phi_L(x) = x$ with $x = \sum_{i=1}^{n} \omega_i(x)$ Both kinds of neurons, threshold and linear, have already been implemented in the past using analog electronic [16], [25].

3.2 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 element 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 $X(i = 1, 2, ..., N)$. 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:

$$X_i = S_{X_i} M_{X_i} 2^{E_{X_i}}$$

Where

- S_{X_i}, designating the sign of X_i 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: $(\frac{1}{2} \leq M_{X_i})\pi 1$);
- 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

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

Where $M_{X_i}^j (j = 1, 2, ..., m)$ 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 $(\frac{1}{2}) \leq M_{X_i}$ implies $M_{X_i}^1 = 1$.

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

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

Where $S_{E_{X_i}} = 1$ is the sign bit of E_{X_i} ($S_{X_i} = 0$ if E_{X_i} is positive or zero, and, $S_{E_{X_i}} = 1$ otherwise), and $E_{X_i} \in \{0, 1\}$ for $0 \leq j \leq p - 1$ and $1 \leq i \leq N$.

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:

$$X_i = (S_{X_i}, M_{X_i}^1, M_{X_i}^2, ..., M_{X_i}^m, S_{E_{X_i}}, E_{X_i}^0, E_{X_i}^1, .., E_{X_i}^{p-2}) (12)$$

The task of finding the k^{th} 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 $X_q(q = 1, 2, ..., N)$.

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

3.3 Order and Selection Networks

The function of the order network $ON_q, (1 \leq q \leq N)$ is to compute the order in the input array X of each element X_q [41]-[53]. The order network ON_q computes the order function (17) and is made up of (N-1) comparison networks, $CN_{i,q}$ $i \in \{1, 2, ..., N\} - \{q\}$, as shown in Fig. 6.

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 , according to the formula

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

Where $a_{(k)}^{0}, a_{(k)}^{1}, ..., a_{(k)}^{n-1}$ 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 $1 \leq q \leq n-1$ and

$1 \leq k \leq N$

The function computed by the EN is defined as

$$eq[ord(X_i, X), k] = \begin{cases} 0, if ord(X_i, X) = k \\ 1, otherwise \end{cases}$$

3.4 The OSF

The OSF is shown in Fig. 12, where the adjustment input Ak determines which order statistic is to appear at the output. The network illustrated in Fig. 11 consists of two kinds of neurons arranged in 11 layers. The number of neurons in OSF for input size N is $14N^2 + (m + p - 9)N + m + p + 2$. There are 11 layers of neurons in the OSF, 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, OSF 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], [72], where the processing time increases along with the number of elements.

The claim that the processing speed of OSF 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 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 OSF to some extent depends on the input size. Even with these limitations, however, the processing speed of the OSF will still be high enough to have the advantage of speed.

Technologies used in OSF implementation are broadly categorized into silicon [19], [39]-[40] using analog, digital, or mixed analog/digital integrated circuits, and optical or electroptical [1], [3]. No matter which medium is used, the performance of the OSF would inevitably be affected by the current level of the medium's technology. Here, we address some problems that might be faced when the OSF is implemented using analog VLSI circuits. Such problems are also common to other neural network models; however, because the OSF has a simple configuration, its implementation is less affected.

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

Whatever technology is utilized, the OSF is not affected by this problem, since the OSF 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 amplifier used in implementing the linear neuron. For some practical applications, this may not be a serious problem [41]-[53].

For example, in image processing applications, the input to the OSF can be easily scaled to fit in the linear range of the amplifier.

4 Simulation and results

The task of the OSF, however, is to select the C or less than C largest D_{ij} values [8], from each column in the HOL input request matrix if there are more than C packets contending for the same time or in the case that there fewer than time respectively. Each input port controller computes de value of D_{ij} of its HOL packets and transmits this value together with the HOL packet's output port destination address to the input interface of the neural network contention controller. Based on the output port destination address, each input interface is affected the equivalent value of D_{ij}.

Let us now describe the interface between the neural network contention controller and the output port controller. With up to C packets being selected from each output port, the packet with the larger D_{ij} value must be transmitted ahead of the packets with lower D_{ij} value. This can be achieved by using the order preserving characteristic of the OSF.

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 D.

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

the elements of input array D.

Let D_{ij} be an element of the input array $D_{ij}(i = 1, 2, ..., N)$

Each element D_{ij} of D is represented in a single way by the triplet $(S_{D_{ij}}, M_{D_{ij}}, E_{D_{ij}})$, as follows: $D_{ij} = S_{D_{ij}} M_{D_{ij}} 2^{E_{D_{ij}}}$

Where

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

The mantissa normalized to m digits $M_{D_{ij}}$, is represented in (binary) base 2 by.

$$M_{D_{ij}} = \sum_{p=1}^{m} M_{D_{ij}}^{p} 2^{-p}$$

Where $M_{D_{ij}}^{p}(p = 1, 2, ..., m)$ are the digits of mantissa $M_{D_{ij}}$ in base 2 $M_{D_{ij}}^{p} \in \{0, 1\}$ for $1 \leq p \leq m$ and $1 \leq i \leq N$.

Agreeing that $\frac{1}{2} \leq M_{D_{ij}}$ implies $M_{D_{ij}}^{1} \neq 0$, i.e, $M_{D_{ij}}^{1} = 1$.

A more efficient implementation of the sorting network is shown in the diagram in fig 12 (a). This sorting network is equivalent to N OSF 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 network" 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 [41]-[53].

A second implementation of the sorting network consists of using n separate OSF networks in parallel, as shown in the diagram in fig 12 (b). Sorting time is fixed and doesn't depend on the size of the input, it is the same time taken for processing a single OSF.

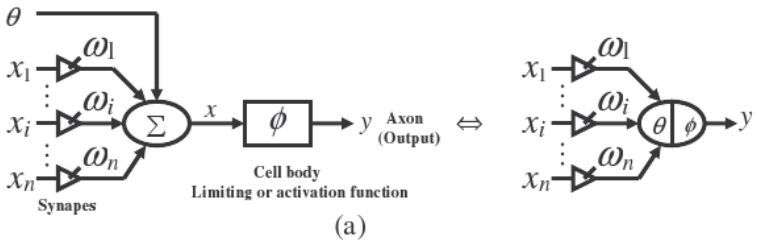
A third implementation is the use of a single OSF network [cf. fig. 12 (c)]. By changing the value of k from 1 to N, the elements of the array will appear at the output sequentially. The advantage is that less neurons are needed; the disadvantage is that the sorting time is proportional to the size of the input.

At each clock pulse [cf. fig. 12(c)], the counter changes state, and k goes from one value to the next. The OSF finally produces at its out put the input array element whose order corresponds to new value of k. the clock frequency must be lower than OSF processing speed, i.e, lower than $(\frac{1}{(11\tau)})$, where τ is the processing time for a single neuron.

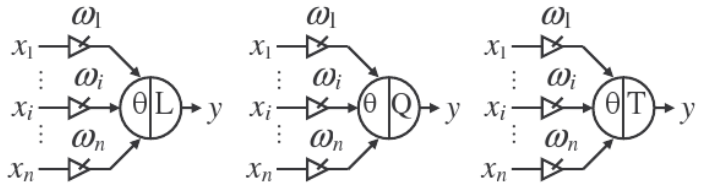
The hardware implementation of the continuous OSF can be designed electronically using the VLSI technology wile the interactive activation OSF network is more suitably implemented through optoelectronic means as the states of the neurons are updated synchronously and discretely.

The simulation is limited to N=5, C=3 and 1 column (each column is computed independently).

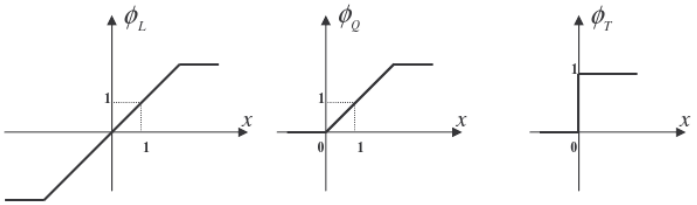
4.1 Simulation and OSF



(a)



(b)



(c)

Figure 2: (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

OSF employs two kinds of neurons, both of which are commonly used in neural network applications. one employs the linear activation function and the other the threshold-logic activation function. Their schematic representations are shown in Fig.2(b), where y is the output. These two kinds of neurons sum the n weighed inputs and pass the result through a nonlinearity according to

$$y = \phi(\sum_{i=1}^{n} \omega_i x_i - \theta)$$

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

The threshold-logic neuron model[see Fig. 2(b)] uses only the binary function [see Fig. 2(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 the equation

$$\phi_T(x) = \begin{cases} 1, & \text{if x} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

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. 12(c)].

in the case of the linear activation function[cf. Fig. 2(b)]. the output y is given by

$$\phi_L(x) = x$$

where ϕ_L is the linear activation function[cf. Fig. 2(c)] defined by $\phi_L(x) = x$ with $x = \sum_{i=1}^{n} \omega_i x_i - \theta$

we introduced 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 $X(i = 1, 2, ..., N)$ 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:

$$X_i = S_{X_i} M_{X_i} 2^{E_{X_i}}$$

where

- S_{X_i} designating the sign of X_i 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

$$M_{X_i} = \sum_{i=1}^{n} M_{X_i}^{j} 2^{E_{X_i}}$$

where $M_{X_i}^{j} (j = 1, 2, ..., m)$ 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"

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

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 $1 \leq i \leq N$.

the exponent E_{X_i}, may be calculated by a single neuron.

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

$$X_i \equiv (S_{X_i}, M^1_{X_i}, M^2_{X_i}, ..., M^m_{X_i}, S_{E_{X_i}}, E^0_{X_i}, E^1_{X_i}, ..., E^{p-2}_{X_i}) \tag{1}$$

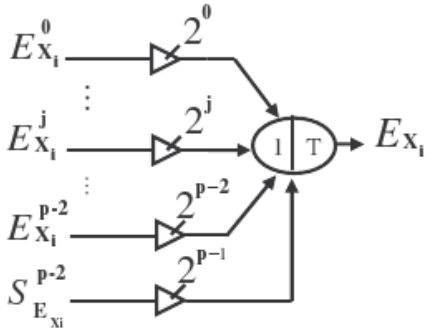


Figure 3: Neural network enabling calculation of exponent $E_{X_i}(1 \leq i \leq N)$

let $\hat{M}_{X_i}$ be the integer associated with $M_{X_i}(i = 1, 2, ..., N)$ according to the formula $\hat{M}_{X_i} = 2^m M_{X_i} = \sum_{j=1}^{m} M^j_{X_i} 2^{m-j}$

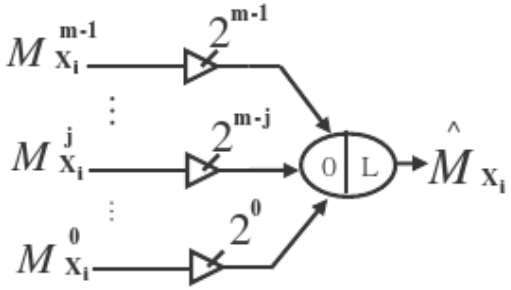


Figure 4: Neural network enabling calculation of $\hat{M}_{X_i}(1 \leq i \leq N)$ according to relationship

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):

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

and

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

if $S_{X_q} \neq 0$ and $S_{X_i} \neq 0$ (i.e. X_q and X_i are not simultaneously positive):

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

and

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

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*

$$ord(X_q,X) = \sum_{i<q} comp(X_i, X_q) + \sum_{i>q} comp(X_i, X_q) \; + \; 1$$

Definition 3. *Let X_k denote the k^{th} largest element of the input array X. Let X_q be an element of the input array X.then*

$$X_q = X_{(k)} \quad iff \quad ord(X_q,X) = k$$

the task of finding the k^{th} 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 $X_q (q = 1, 2, ..., N)$.
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.

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

$$X_i = M_{X_i} 2^{E_{X_i}}$$

$\mid X_i \mid$ can be represented as an (m+p) bit binary number

$$\mid X_i \mid \equiv (M^1_{X_i}, M^2_{X_i}, ..., M^m_{X_i}, S_{E_{X_i}}, E^0_{X_i}, E^1_{X_i}, ..., E^{p-2}_{X_i})$$

Definition 4. *Let $\mid X_i \mid$ and $\mid X_q \mid$ be the absolute value of X_i and X_q, respectively; we say that $\mid X_q \mid > \mid X_i \mid$ if and only if*

1. $E_{X_q} > E_{X_i}$ *or*
2. $E_{X_q} = E_{X_i}$ *and* $\hat{M}_{X_q} > \hat{M}_{X_i}$

Definition 5. *Let $\mid X_i \mid$ and $\mid X_q \mid$ be the absolute value of X_i and X_q, respectively. Then*

$$comp(\mid X_i \mid, \mid X_q \mid) = \begin{cases} S_1, & \text{if } i < q \ ; \\ S_2, & \text{if } i > q \ . \end{cases}$$

where

$$S_1 = \phi_T[\phi_T(E_{X_q} - E_{X_i} - 1) - \phi_T(E_{X_i} - E_{X_q} - 1) + \phi_T(-\phi_T(E_{X_q} - E_{X_i} - 1) -$$

$$\phi_T(E_{X_i} - E_{X_q} - 1) + \phi_T(\hat{M}_{X_q} - \hat{M}_{X_i} - 1) - 1) - \phi_T(-\phi_T(E_{X_q} - E_{X_i} - 1) -$$

$$\phi_T(E_{X_i} - E_{X_q} - 1) + \phi_T(\hat{M}_{X_i} - \hat{M}_{X_q} - 1) - 1) + \phi_T(\phi_T(-\phi_T(\hat{M}_{X_q} - \hat{M}_{X_i} - 1) -$$

$$\phi_T(\hat{M}_{X_i} - \hat{M}_{X_q} - 1)) - \phi_T(E_{X_q} - E_{X_i} - 1) - \phi_T(E_{X_i} - E_{X_q} - 1) - 1)]$$

$$S_2 = \phi_T[\phi_T(E_{X_q} - E_{X_i} - 1) - \phi_T(E_{X_i} - E_{X_q} - 1) + \phi_T(-\phi_T(E_{X_q} - E_{X_i} - 1) -$$

$$\phi_T(E_{X_i} - E_{X_q} - 1) + \phi_T(\hat{M}_{X_q} - \hat{M}_{X_i} - 1) - 1) - \phi_T(-\phi_T(E_{X_q} - E_{X_i} - 1) -$$

$$\phi_T(E_{X_i} - E_{X_q} - 1) + \phi_T(\hat{M}_{X_i} - \hat{M}_{X_q} - 1) - 1) - 1)]$$

Definition 6. *Let $\mid X_i \mid$ and $\mid X_q \mid$ be the absolute value of X_i and X_q, respectively; we say that $\mid X_q \mid > \mid X_i \mid$ if and only if*

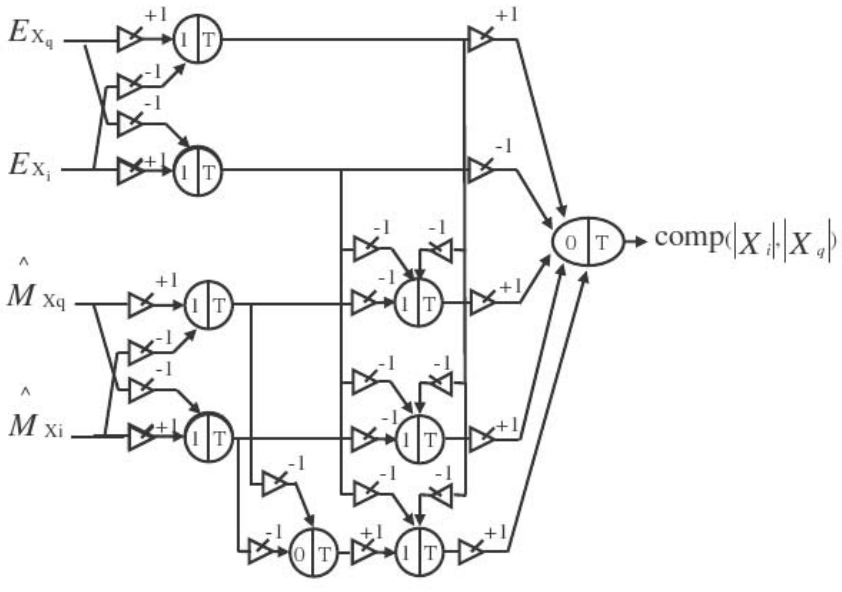
1. $S_{X_q} = 0$ *and* $S_{X_q} = 1$ *or*

2. $S_{X_q} = 0$ *and* $S_{X_q} = 0$ *and* $\mid X_q \mid > \mid X_i \mid$ *or*

3. $S_{X_q} = 1$ *and* $S_{X_i} = 1$ *and* $\mid X_q \mid < \mid X_i \mid$

Definition 7. *Let* $\mid X_i \mid$ *and* $\mid X_q \mid$ *be the absolute value of* X_i *and* X_q*, respectively. then*

$$comp(X_i, X_q) = \phi_T(-S_{X_q} - S_{X_i} + comp(\mid X_i \mid, \mid X_q \mid) - 1) +$$
$$\phi_T(S_{X_q} + S_{X_i} - comp(\mid X_i \mid, \mid X_q \mid) - 2) + \phi_T(-S_{X_q} + S_{X_i} - 1)$$



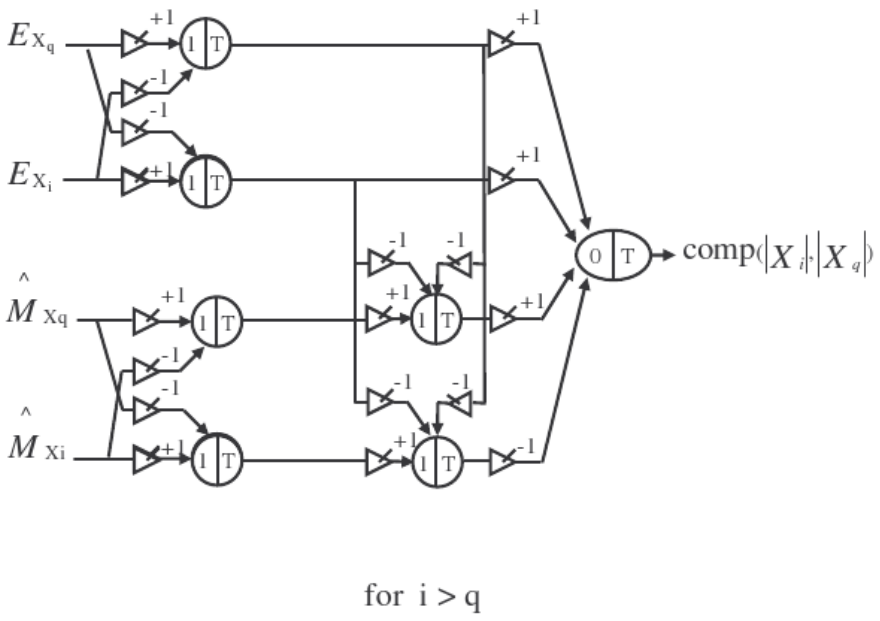


Figure 5: Neural network, for computing the function comparison

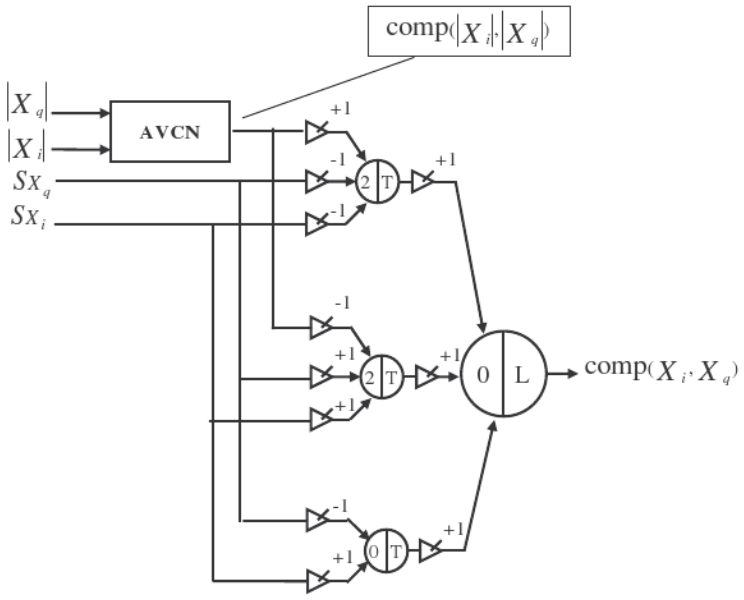


Figure 6: Comparison network of X_i and X_q, CN(i,q), where AVCN is the network depicted

The function of the order network $ON_q(1 \leq q \leq N)$, is to compute the order in the input array X of each element X_q and is made up of(N-1) comparison networks, $CN(i,q)i \in \{1,2,...,N\} - \{q\}$.

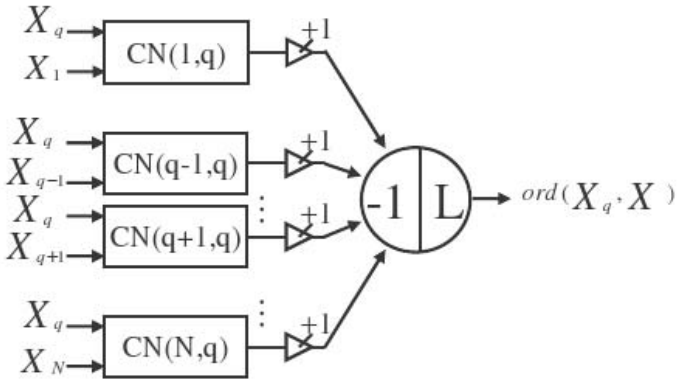


Figure 7: Order network of $ON_q(1 \leq q \leq N)$, where $CN_{i,q}, i \in \{1,2,...,N\} - \{q\}$ is the network depicted

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 network(ENs) and a detection network(DN):

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 the formula

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

where $a_{(k)}^0, a_{(k)}^1, ..., a_{(k)}^{n-1}$ 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

$$eq[ord(X_i, X), k] = \begin{cases} 0, & \text{if ord}(X_i,\text{X}) = \text{k} \\ 1, & \text{otherwise.} \end{cases}$$

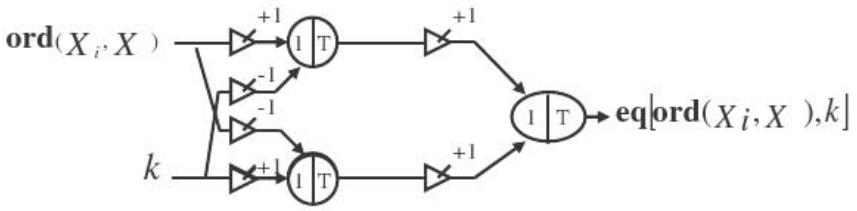


Figure 8: Order network of $ON_q(1 \leq q \leq N)$, where $CN_{i,q}, i \in \{1, 2, ..., N\} - \{q\}$ is the network depicted

Proposition 1.

$$eq[ord(X_i, X), k] = \phi_T(\phi_T(ord(X_i, X) - k - 1) + \phi_T(k - ord(X_i, X) - 1) - 1)$$

$$for \;\; 1 \leq i \leq N \;\; and \;\; 1 \leq k \leq N$$

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 2. *Let the equation at the bottom of the page [relationship(1)] be the kth largest element of input array X. Then*

$$S_{X_{(k)}} = f(S_{X_1}, S_{X_2}, ..., S_{X_N}) \quad (3)$$

$$M^j_{X_{(k)}} = f(M^j_{X_1}, M^j_{X_2}, ..., M^j_{X_N}), \quad for \;\; 1 \leq j \leq m \quad (4)$$

$$S_{E_{X_{(k)}}} = f(S_{E_{X_1}}, S_{E_{X_2}}, ..., S_{E_{X_N}}) \quad (5)$$

$$E^q_{X_{(k)}} = f(E^q_{X_1}, E^q_{X_2}, ..., E^q_{X_N}), \quad for \;\; 0 \leq q \leq p - 2 \quad (6)$$

where
$f(b_1, b2, ..., b_N) = \phi_T(\sum_{i=1}^{N} \phi_T(b_i - eq[ord(X_i, X), k] - 1) - 1)$
with
$b_i = (S_{X_i}$ or $M^j_{X_i} (= 1, 2, ..., m)$ or $S_{E_{X_i}}$ or $E_{X_i} (q = 0, 1, ..., p - 2))$
for $i = 1, 2, ..., N$.
Function (30)-(33) are computed by the network by the diagram:

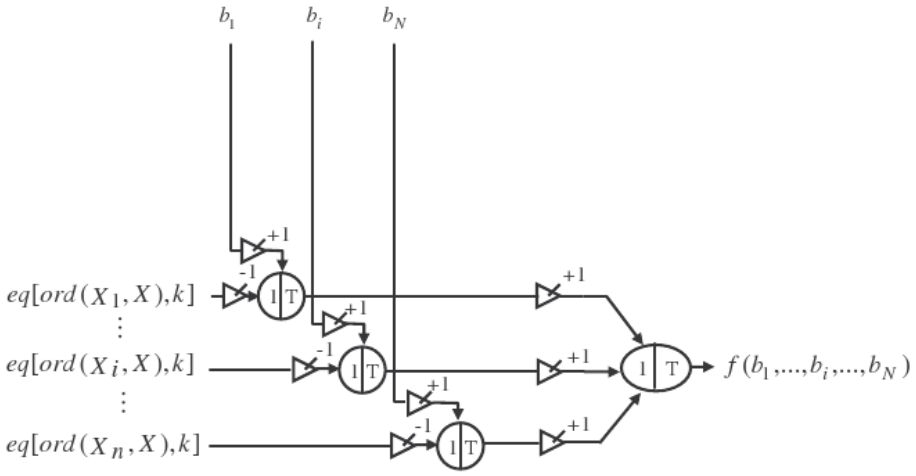


Figure 9: Detection network DN

The selection network SN transfers only the appropriate element whose order equals k to the output:

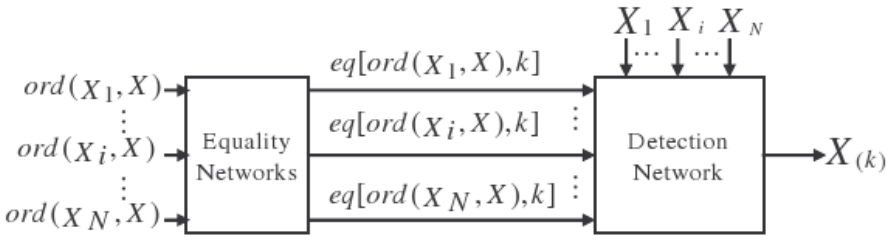


Figure 10: Selection network DN

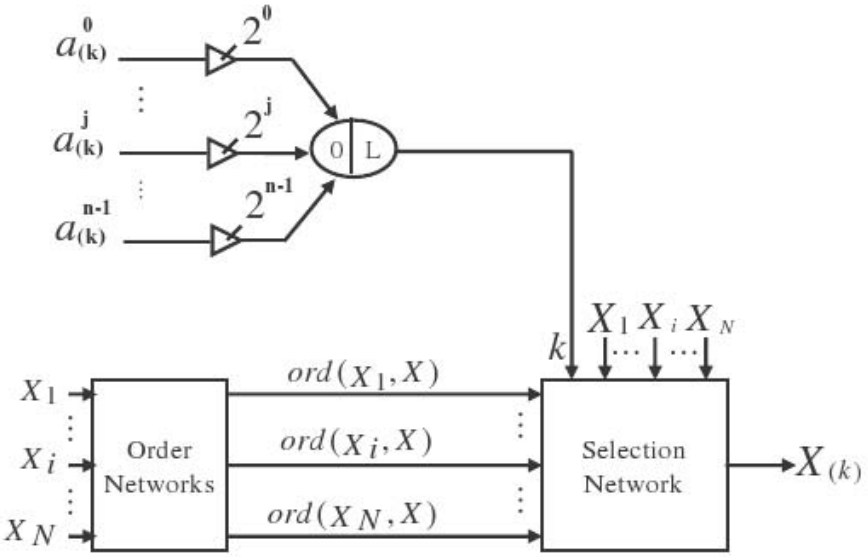


Figure 11: Adjustable order statistic filter OSF, where $(a^0_{(k)}, ..., a^j_{(k)}, ..., a^{n-1}_{(k)})$ is the word of the command given by the relationship (2)

The OSF is shown in Fig.10, where the adjustment input A_k determines which order statistic is to appear at the output. The network illustrated in Fig.10 consists of two kinds of neurons arranged in 11 layers. The number of neurons in OSF for input size N is $14N^2 + (m + p - 9)N + m + p + 2$. There are 11 layers of neurons in the OSF, 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, OSF'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,where the processing time increases along with the number of elements. The claim that the processing speed of OSF 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

OSF to some extent depends on the input size. Even with these limitations, however, the processing speed of the OSF will still be high enough to have the advantage of speed.

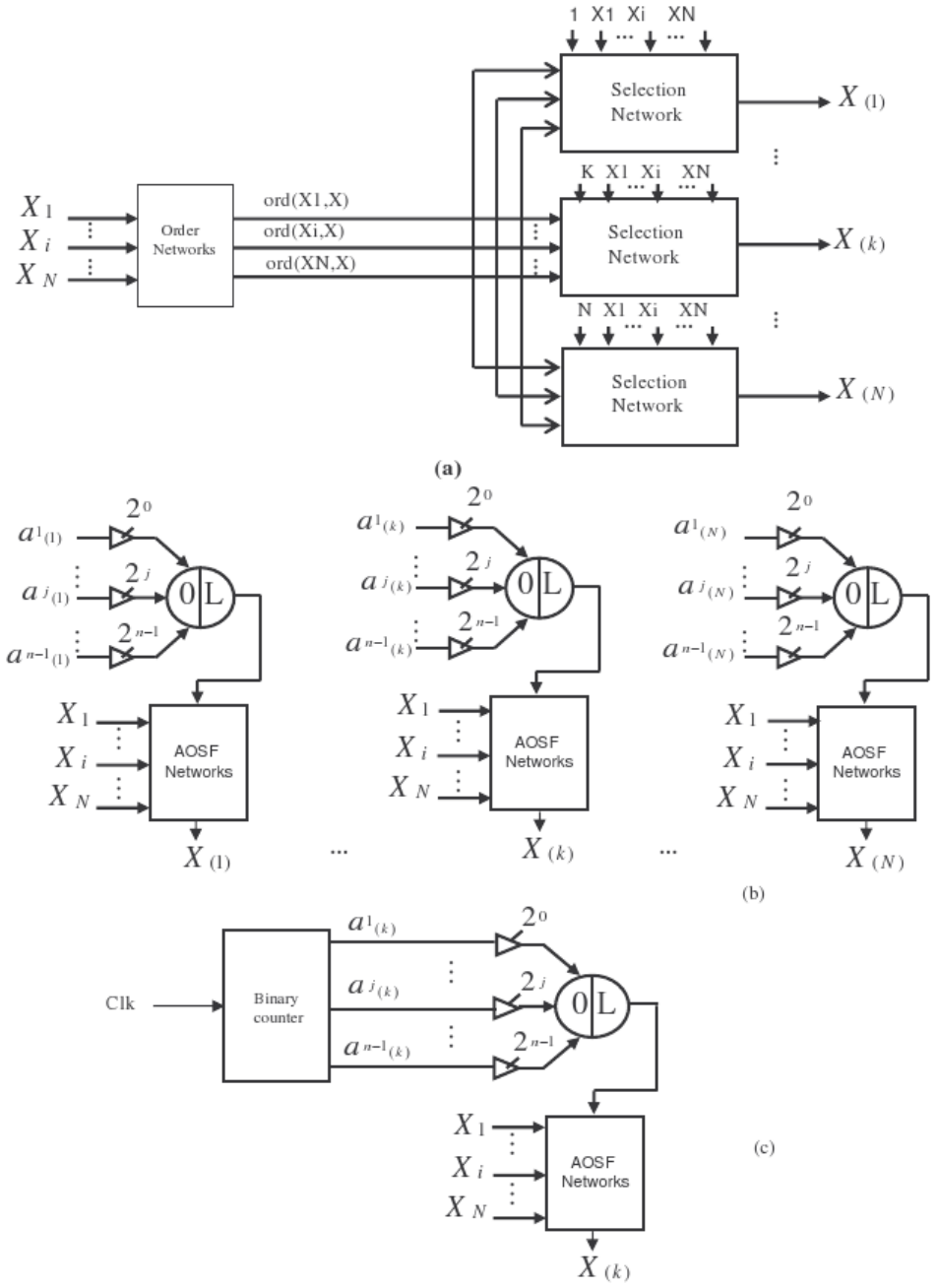


Figure 12: (a) Sorting network made up of N OSF networks in parallel whose common module "order networks" has been merged (b)Sorting network made up of N separate OSF networks in parallel. (c)Sequential sorting network made up of a single OSF network

4.2 Simulation and MATLAB

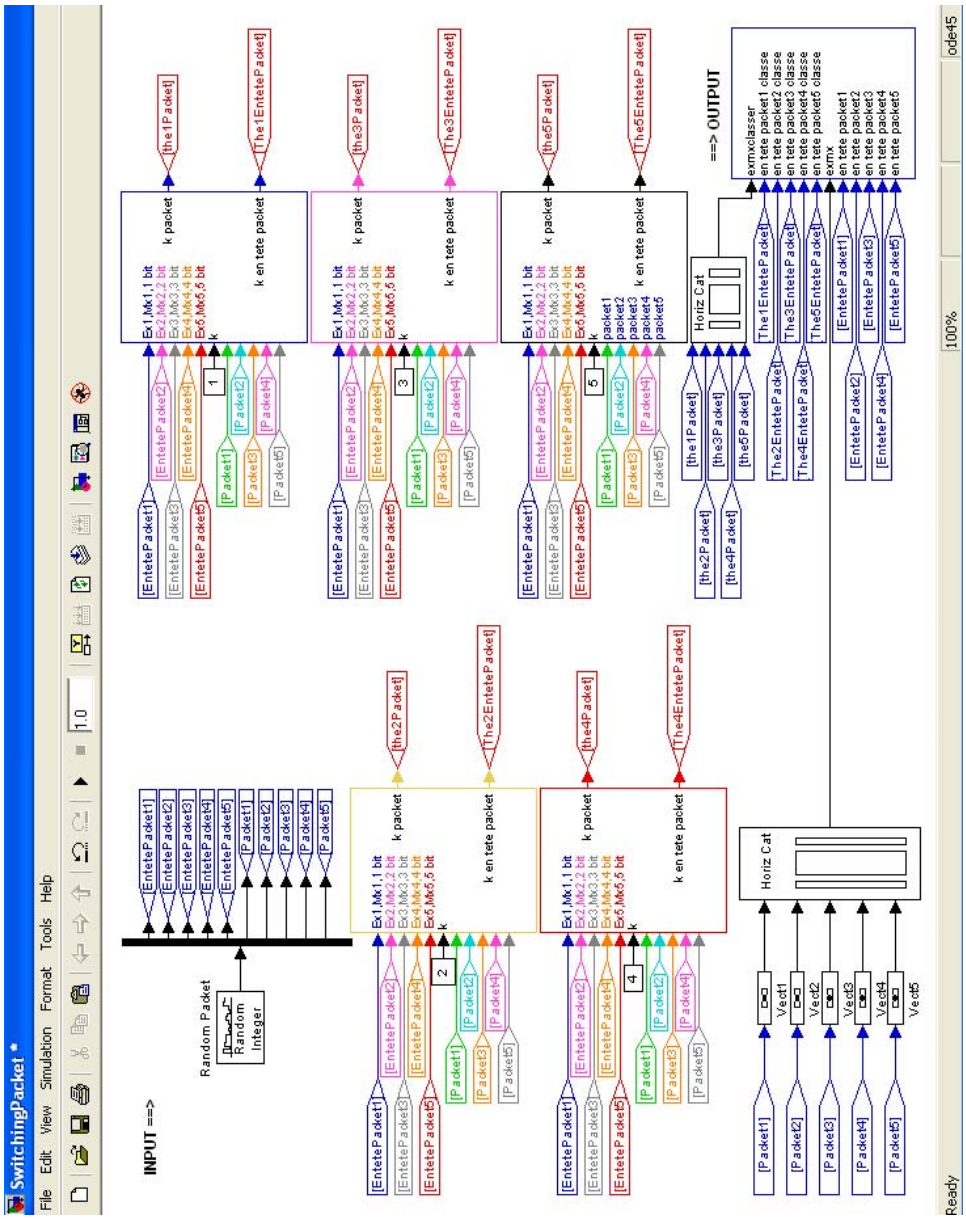


Figure 13: Home Page For Simulation Simulink

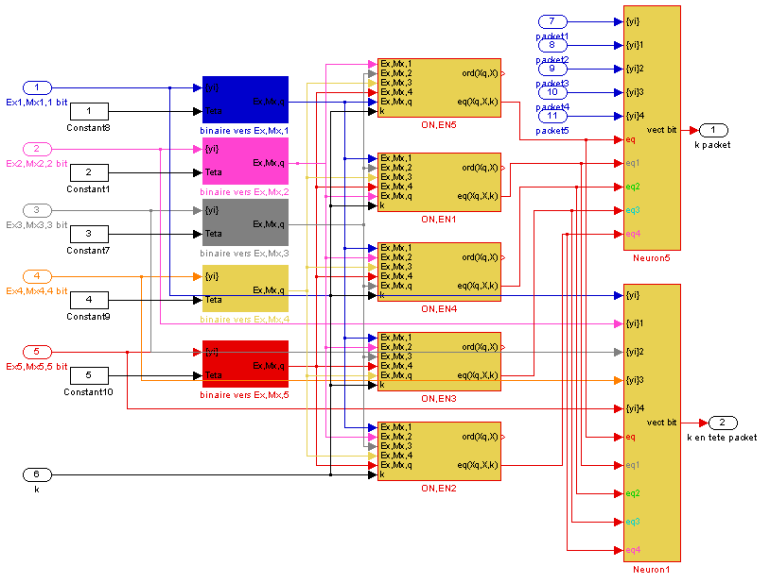


Figure 14: ONSN Simulation Simulink

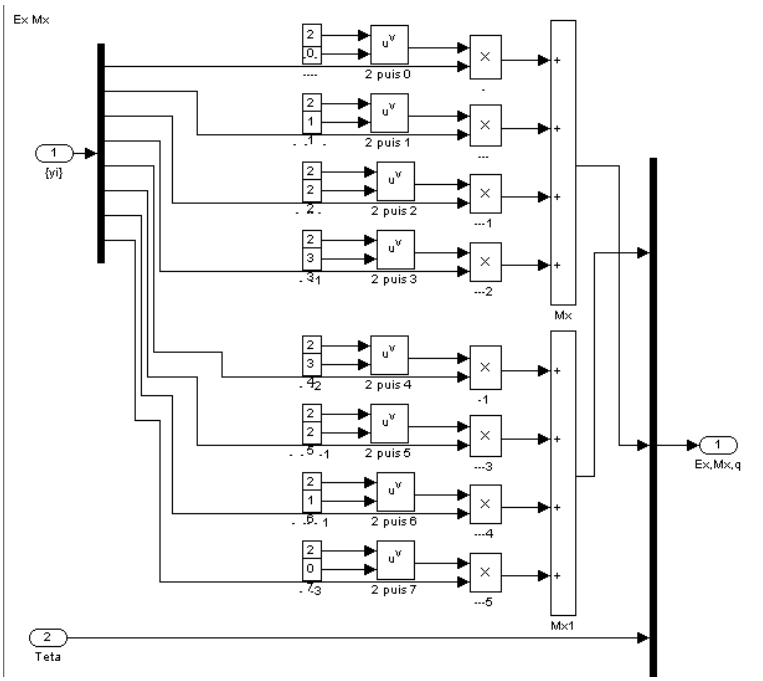


Figure 15: Calculate of Ex and Mx For Simulation Simulink

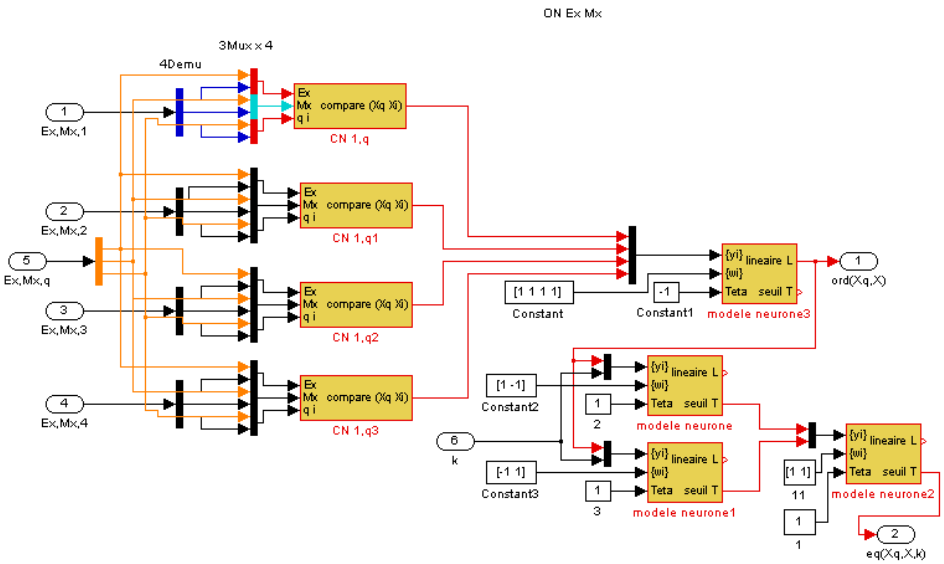


Figure 16: Order Network Xq For Simulation Simulink

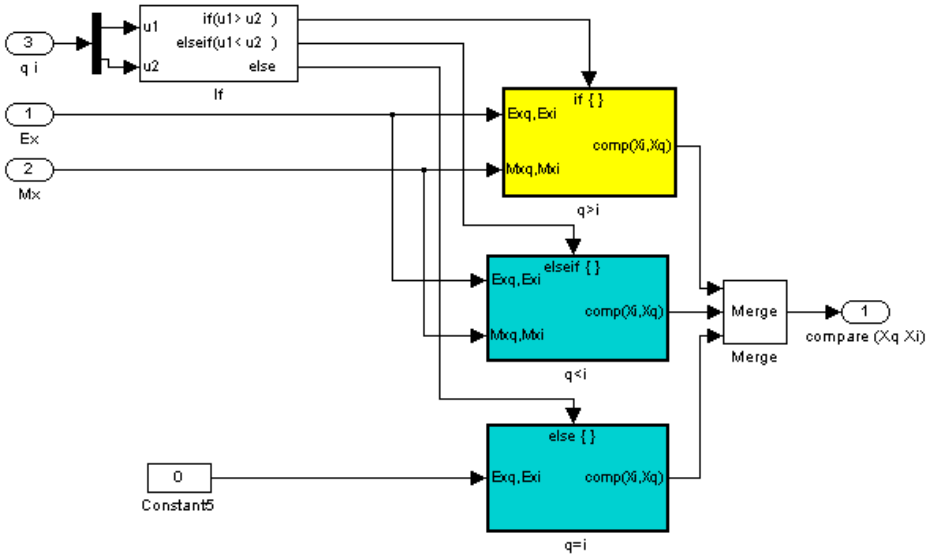


Figure 17: Compare and equal Network For Simulation Simulink

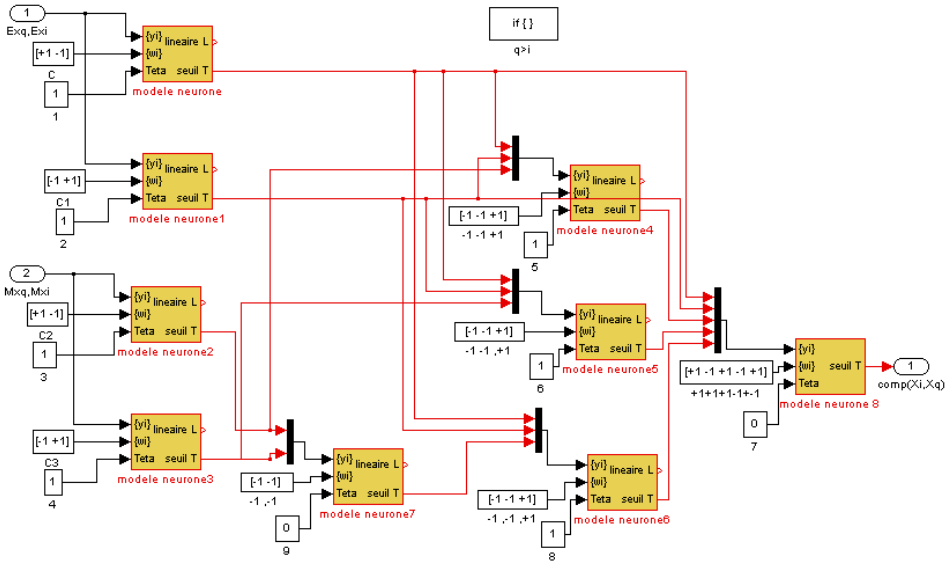


Figure 18: Compare Network For Simulation Simulink

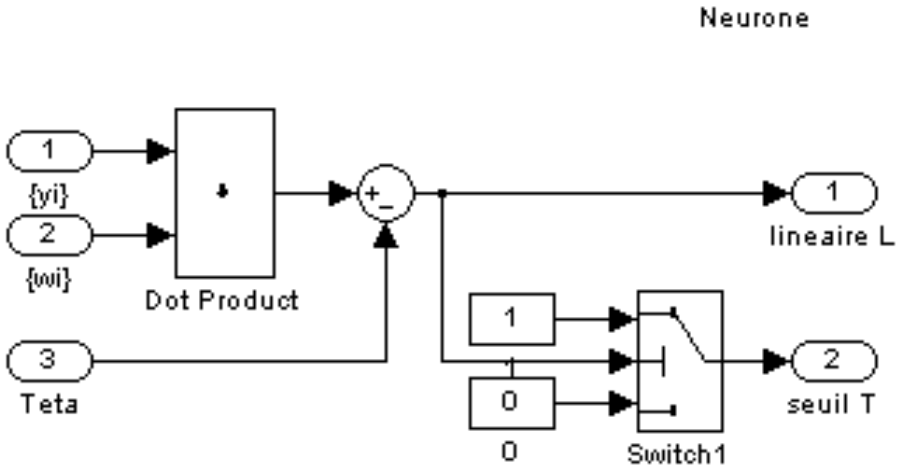


Figure 19: Neuron For Simulation Simulink

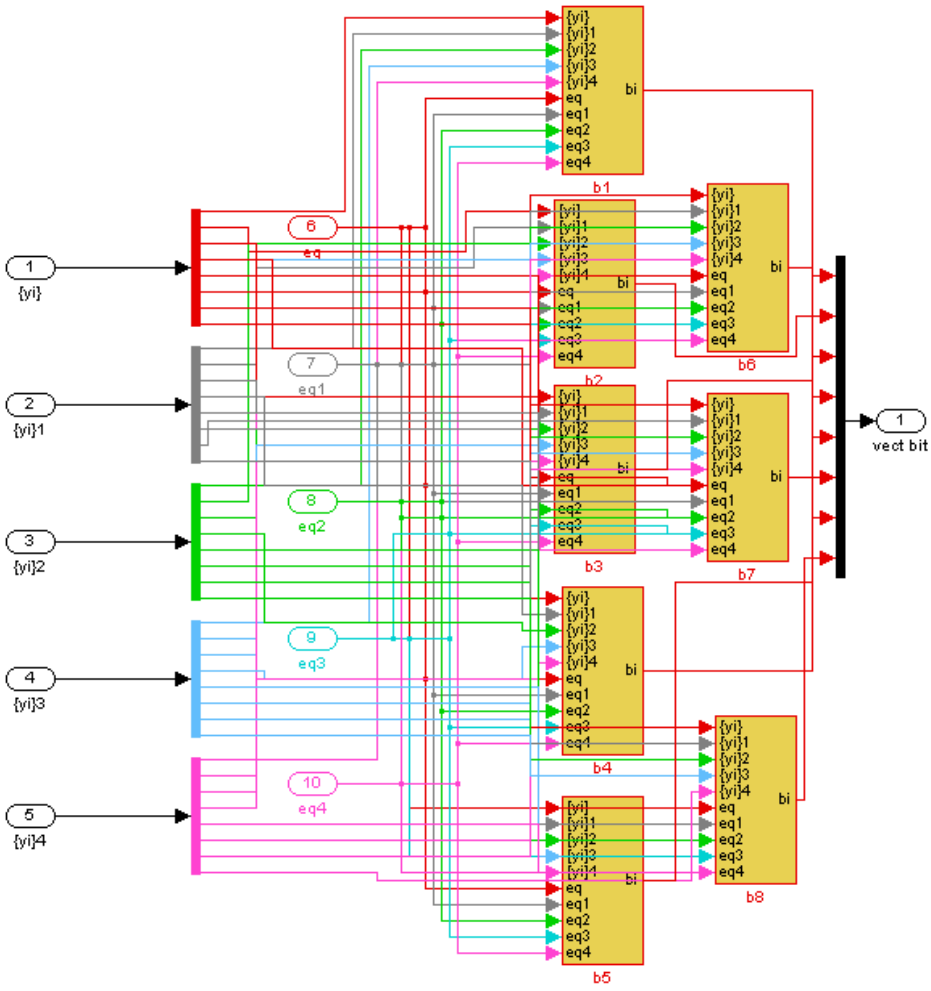


Figure 20: K Packet For Simulation Simulink

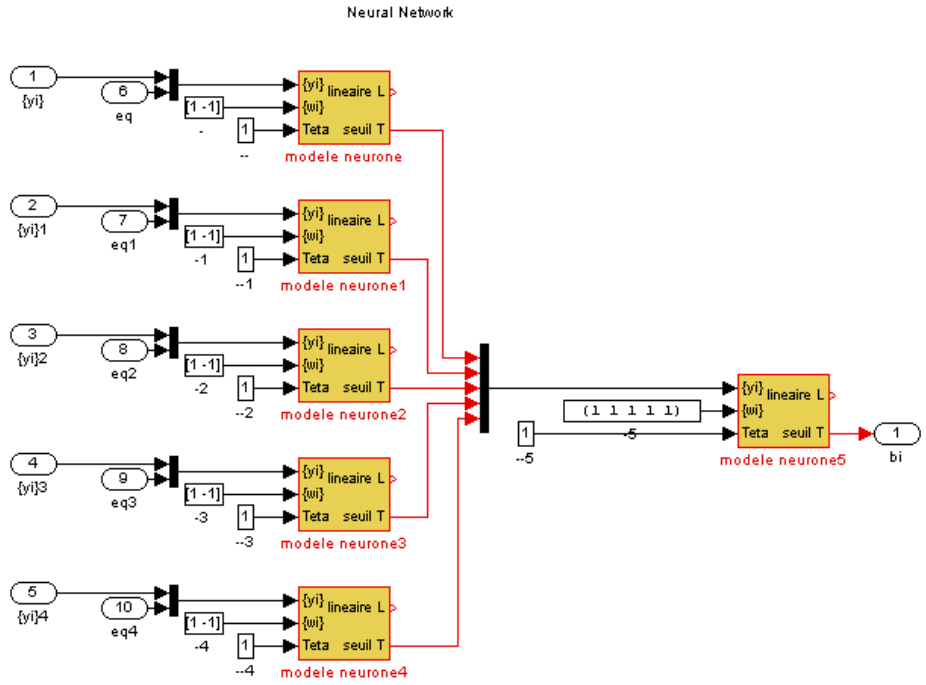


Figure 21: Neural Network For Simulation Simulink

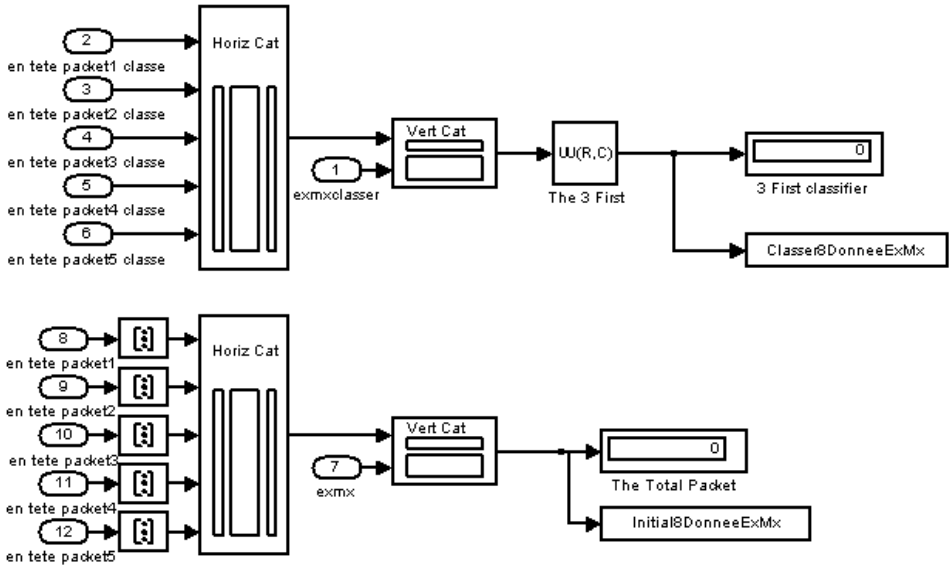


Figure 22: OUTPUT For Simulation Simulink

Packet	Ex/Mx	Ex and Mx (Binary)
"1"	14/2	0 1 1 1 0 0 1 0 0 0 1 0 0 1 0 0
"2"	1/14	1 0 0 0 1 1 1 0 0 1 0 0 1 0 1 0
"3"	12/13	0 0 1 1 1 1 0 1 0 0 0 0 1 1 0 1
"4"	0/14	0 0 0 0 1 1 1 0 1 1 1 0 0 0 1 0
"5"	3/15	1 1 0 0 1 1 1 1 1 0 1 1 0 0 1 0

TABLE 1: Initial state: t = 0, N=5, C=3, 1 column, and STEP-SIZE= τ.

Packet	Ex/Mx	Ex and Mx (Binary)
"5"	3/15	1 1 0 0 1 1 1 1 1 0 1 1 0 0 1 0
"2"	1/14	1 0 0 0 1 1 1 0 0 1 0 0 1 0 1 0
"4"	0/14	0 0 0 0 1 1 1 0 1 1 1 0 0 0 1 0

TABLE 2: Final state: t=11τ, N=5, C=3, 1 column, and STEP-SIZE= τ.

From the simulation results presented above, we can deduce the following conclusions concerning the criteria required in OSF:

- 1- Results obtained 100%.
- 2- Hardware implementation introduces no error.
- 3- The results are computed in a fixed time, independently of the number of input, output and the value of C.
- 4- The OSF is modular, so we can make extension, without changing the computing time (fixed).

5 Conclusion

We have demonstrate that the contention controller based on the OSF neural network can be used for real time computation of a no blocking switching configuration for a packet switch with a speed up factor C. With the approach employed here for the conception of OSF, 100% accurate results may be obtained, knowing that the hardware implementation introduces no error. For many engineering applications such a high level of accuracy is required. Unfortunately this level of accuracy is hard to achieve by using the learning approach even under the assumption of error-free hardware implementation. Unlike A Neural-Network Contention Controller for Packet Switching Networks [8], the OSF neural network processed in a fixed time

OSF has very simple configuration, which makes it less subject to the problem caused poor absolute accuracy in setting up values of the connection

weights. In OSF the connection weights between the neurons 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 the connection. As well as this, the problem due to saturation characteristics of the amplifiers used in implementing the linear and the quasi-linear neurons does not limit OSF performance as in most practical applications the input to OSF can be easily scaled to fit in the linear range of the amplifiers.

The compatibility of the OSF network with the approach taken to resolve the output contention as described in section II.2, makes it possible to implement a wide variety of service selection policies, unlike the Batcher sorting network [18] and the knockout contention controller [76]. The OSF neural network allows for a wide variety of service selection policies to be used, for example; random order, longest queue, FIFO, or priority service selection. This is important in the voice and video will be clearly delay sensitive compared with data type services. It is therefore necessary for the contention controller to implement a priority service selection policy to support such services.

The significance of the proposed OSF neural network contention controller is that it does not suffer the packet loss as compared with other techniques such as Hopfield neural network and the winner-take-all circuits with overlapping and bypass input queuing. The preservation of the FIFO input sequence is a superior property of our network and minimize the complexity of network implementation.

The OSF neural network contention controller proposed here would play a major development optical packet switching, all optical networks, MPLS.

References

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

[2] 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.

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

[4] 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.

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

[6] Bens, V. E., Mathematical Theory of connecting Networks and Telephone Traffic. Academic Press, New York, 1965.

[7] L. Berger et. Al (2003, Jan). RFC 3471- Generalized multi-protocol label switching (GMPLS) signalling functional description. Internet Engineering Task Force (IETF) proposed Standard.

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

[10] 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.

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

[12] T. X. Brown,"Neural networks for switching," in Neural Networks in Telecommunications, B. Yuhas and N. Ansari, Eds. Boston, MA: Kluwer, 1994.

[13] A. C. Bovic, 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.

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

[17] 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.

[18] J. J. Degan, G. W. R. Luderer, and A. K. Vaidya, "Fast packet technology for future switches, " AT&T Tech. J., pp. 36-50, Mar./Apr. 1989.

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

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

[21] R. Erlansan 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.

[22] K. Fukushima, "A neural network for visual pattern recognition," IEEE Comput, vol. 21, pp. 65-75, Mar. 1988.

[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] 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.

[27] Inose, H., An Introduction to Digital Integrated Communications Systems, Univ. Tokyo Press, 1979.

[28] 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.

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

[30] S. Y. Kim, S. H. Lee, S. S. Lee and J. S. Lee, "Upgading WDM networks using ultradense WDM channel group," Photonics Technology Letters, IEEE, vol. 16 no.8, pp. 1966-1968, Aug., 2004.

[31] T. Kohonen, "Correlation matrix memories," IEEE Trans. Comput., vol. C-12, pp. 353-359, 1972.

[32] 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 System, D. S. Touretzky, Ed. Los Altos, CA: Morgan Kaufmann, 1989, vol. pp. 703-711.

[33] 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.

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

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

[36] 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.

[37] R. P. Lippman, "An introduction to computing with neural nets," IEEE Trans. Acoust., Speech, Signal Processing, vol. 35, pp. 2_44, Apr. 1987.

[38] A. Marrakchi and T. Troudet, "A neural net arbitrator for large crossbar packet switches," IEEE Trans. Circuits Syst., vol. 36, no. 7, p. 1039-1041, July 1989.

[39] C. Mead and M. Ismail, Analog VLSI and Neural Systems. Reading, MA: Addison-Wesley, 1989.

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

[41] M. Mestari and A.NAmir, "AMAXNET: A neural network implementation of adjustable MAXNET in fixed time," in Proc. IFAC-IFIP-IMACS Int. Conf. Control Industriel System, vol. 2, Belfort, France, May 20-22, 1997, pp. 543-549.

[42] M. Mestari and A.Namir, "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.

[43] M. Mestari and A. Namir, "AOSNET: A neural network implementation of adjustable order statistic filters in fixed time, "SIAMS J., vol. 36, pp. $509_5$35, 2000.

[44] M. Mestari and A. Namir, "A neural network implementation of L∞, metric partitional clustering in fixed time, "SIAMS J., vol. 41, no. 2, pp. 351-380, 2001.

[45] 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.

[46] M. Mestari, "A Analog Neural Network Implementation in Fixed Time of Adjustable Order Statistic Filters and Applications", IEEE Transactions on Neural networks, vol. 15, no. 3, May 2004, pp. 766-785.

[47] M. Mestari, M. Benzirar, M. Elhammouti, A. Yeznasni, and J. Meziane " Calculation of the diffusion coefficient in a heated turbulant medium using optimisation technique", International Journal of Engineering Science, vol. 40, pp. 2001-2021, 2002.

[48] M. Mestari, M. Benzirar, M. Elhammouti, A. Yaznasni, and J. Meziane "Measurement of inverse temperature gradient along a turbulent plane flame using an optical method", communications in Nonlinear Science

and Numeric Simulation Journal, vol. 9, pp. 367-377, 2004.

[49] M. Mestari and A. Namir, "Optimal control of discrete nonlinear dynamic system using decomposition coordination method II", IFIP Proc. Conf. on Optimization-Based Computer-Aided Modelling and Design, 28-30 May, Noisy-le-grand, Paris, (France), vol. 1, pp. 25.1-25.4, 1996.

[50] M. Mestari and A. Namir, "Optimisation control of discrete nonlinear dynamic system using decomposition coordination method I", AMSE Proc. Internat. Conf. on Communication, Signal and Systems, vol. 2, pp. 873-882, 1996.

[51] M. Mestari, K. Akodadi and A. Badi, "Neural networks for solving nonlinear constrained multi-objective optimization problems", IEEE-URST-CST Proc. Internat. Sympasium TELECOM'2005, March 23-25 Rabat (Morocco), pp. 687-690, 2005.

[52] M. Mestari, K. Akodadi and A. Badi, "Neural networks classifier", IEEE-URST-CST Proc. Internat. Sympasium TELECOM'2005, March 23-25 Rabat (Morocco), pp. 687-690, 2005.

[53] M. Mestari, A. Namir, K. Akodadi, and A. Badi "O(1) Time neural network distance classifier", MCSEAI'06, Dec 07-09 Agadir (Morocco), 2006.

[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, "sub-logarithmic time sorting algorithm on a reconfigurable artay," IEICE, vol.E-74, No.11.pp.3894-3901,Nov.1991.

[57] 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. configurable artay, "IEICE, vol. E-74, no. 11, pp. 3894-3901, Nov. 1991.

[58] K. Nichols et al. (2001, apr). RFC 3068- definition of differentiated services per domain behaviours and rules for their specification. Internet Engineering Task Force (IEFTF) Proposed Standard.

[59] M. Nigam and S. Sahni, "sorting n numbers on nxn reconfigurable meshes with buses, "in Proc. Int. Parallel Processing Symp., Apr. 1993, pp. 174-181.

[60] J. Pankove, C. Radehaus, and K. Wanger, "Winner-take-all neural net with memory, "Electron. Lent, vol. 26, no. 6, pp. 349-350, Mar. 1990.

[61] V. K. Prasana Kumar and C. S. Ragavenda, "Array processor with multiple broadcasting," J. Parallel Distrib. Comput. Vol. 4, pp. 173-190, 1987.

[62] Pattavina, A, "Architectures and performance of optical packet switching nodes for IP networks," Journal of lightwave technology, vol. 23, pp. 1023-1032, (3), March, 2005.

[63] D. S. Richrad, "VLSI median filters," IEEE Trans. Acoust. Speech, Signal Processing, vol, 38, pp. 145-153, Jan. 1990.

[64] J. Rothsten, "Bus automata, brains, and mental model," IEEE Trans. Syst., Man, Cybern., vol. 18, pp. 522-531, Apr. 1988.

[65] Schwartz, Telecommunication Networks: Protocols, Modelling, and Analysis, Addison-Wesley Pub. Co., Reading, MA., 1979.

[66] G. Seiler and J. A. Nossek, "Winner-take-all cellular neural networks," IEEE Trans. Circuits Syst. II, vol. 40, pp. 184-194, Mar. 1993.

[67] P. Shi and R. K. Ward, "A neural network implementation of median filtering," in IEEE Pacific Rim Conf., Victoria, BC, Canada, 1989.

[68] Shun Yao, S.J B, Mukherjee B, Dixit S." All optical packet switching for metropolitan area networks: opportunities and challenges" Communications Magazine IEEE, vol.39, no, pp. 142-148, March 2001.

[69] L. Snyder, "Introduction to the reconfigurable highly parallel computer," Comput., vol. 15, no. 1, pp. 47-56, Jan. 1982.

[70] B. W. Suter and M. Kabrisky, "On a magnitude preserving iterative Maxnet algorithm," neural Comput., vol. 4, pp. 224-233, 1992.

[71] T. P. Troudet and S. M. Walters, "Neural network architecture for crossbar switch control," IEEE Trans. Circuits Syst., vol 38, no. 1, pp. 42-56, Jan. 1991.

[72] 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.

[73] B. F. Wang, G. H. Chen, and F. C. Lin, "Constant time sorting ona processor array with a reconfigurable bus systems, "Inform. Procesing Lett., pp. 187-192, 1990.

[74] J. H. Winters and C. Rose, "Minimum distance automata in parallel networks for optimum classification," Neural Networks, vol. 2, pp. 127-132, 1989.

[75] W. J. Wolfe, D. mathis, C. Anderson, J. Rothman, M. Gottler, G. Brady, R. Walker, G. Duane, and G. Alaghband, "K-winer networks," IEEE trans. Neural Networks, vol. 2, no. 2, pp. 310-315, Mar. 1991.

[76] Y. S. Yeh, M. G. Hluchyj, and A. S. Acompora, "The knockout switch," IEEE J. Select. Areas Commun., vol. SAC-5, no. 8, pp. 1274-1283, Oct. 1987.

[77] S. Zunino, "Circuit implementation of the K-winer machine," Electron. Lett., vol. 35, no. 14, pp. 1172-1173, July 8, 1999.

Received: November, 2008