A Two Step Sorting Algorithm

Yoshiyasu Takefuji*, Kuo-Chun Lee*, and Toshimitsu Tanaka**

*Department of Electrical Engineering
Center for Automation and Intelligent Systems Research
Case Western Reserve University
Cleveland, Ohio 44106

**Sumitomo Metal Industries , Ltd.
Kashima Steel Works, Ibaragi 314 Japan

Abstract

A new parallel algorithm based on neural networks for solving sorting problems is presented in this paper. The proposed algorithm uses $O(n^2)$ processing elements called binary neuron where n is the unsorted elements. It requires two and only two steps, while the conventional parallel sorting algorithm using O(n) processors proposed by Leighton needs the computation time O(log n) [1]. A set of simulation results substantiates the proposed algorithm. The hardware system based on the proposed parallel algorithm is also presented in this paper.

1. Neural network representation

Processing elements used in the new algorithm are called binary neurons where they perform the function of a simplified biological neuron. The binary neurons have been successfully used for solving graph planarization problems [2] and tiling problems [3]. The output of the binary neurons is given by:

$$V_i=f(U_i)= \begin{cases} 1 & \text{if } U_i > 0 \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

where V_i is the output of the ith neuron and U_i is the input to the ith neuron. The unsorted n-1 positive integers, N_1, N_2, ..., N_{n-1}, and 0 (zero) which is a dummy (minimum) number are given where the subscript i indicates the location of the register i which contains the number N_i. The goal of sorting is to find a permutation $\Pi=(\pi_1, \pi_2, ..., \pi_{n-1})$ such that $0<N_{\pi_1}<N_{\pi_2}<...<N_{\pi_{n-1}}$. An n×n neural network array is provided where each row and column corresponds to the location of the register . The n×n array represents the directed adjacency matrix where one and only one neuron in

the ith row (i=1,...,n) will be fired in order to determine the sorting order between N_{π_i} and $N_{\pi_{i+1}}$ as shown in Fig. 1. Note that $N_{\pi_{i+1}}$ must be greater than N_{π_i} but it must be the nearest number to N_{π_i}. The motion equation of the XYth neuron which is in the Xth row and in the Yth column (X=1,...,n Y=1,...,n X≠Y) is given by:

$$\frac{dU_{XY}}{dt} = -g(N_X, N_Y)\left(\sum_{i \neq X}^{n} g(N_i, N_Y)\, V_{Xi} - 1\right) \qquad (2)$$

$$\text{where } g(L, R) = \begin{cases} 1 & \text{if } L \leq R \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

It is very important that all of the initial values at t=0 for U_{XY}(X=1,...,n Y=1,...,n X≠Y) are set to zero so that all of the V_{XY}(X=1,...,n Y=1,...,n X≠Y) are zeros, because of Eq. (1). If N_Y is greater equal N_X, Eq. (2) at t=0 becomes one because V_{XY} are zeros; and the neuron will fire. If N_Y is less than N_X, Eq. (2) will be negative so that the neuron is not fired. It is concluded that at t=0 the zero initial values of U_{XY} forces the neurons whose number is greater than N_X to fire as shown in Fig.1(a). The term $(\Sigma g(N_i,N_Y)V_{Xi} -1)$ forces one and only one neuron to fire per row. The function of $g(N_i,N_Y)$ plays a key role in our method to determine which permutation connection should remain and which should be removed. After the first iteration, one and only one neuron among fired neurons whose number must be not only greater than N_X but also be the smallest number is forced to remain fired. In other words, only if N_Y is greater equal N_X and the smallest number among N_i (i=1,...,n-1), the term $\Sigma g(N_i,N_Y)V_{Xi}$ is one so that Eq. (2) will be zero and the neuron is forced to remain fired, otherwise $\Sigma g(N_i,N_Y)V_{Xi}$ is more than one so that Eq. (2) will be negative and the fired neuron should be unfired, as shown in Fig.1 (b).

2. Algorithm and the simulation result

The first order Euler method was applied to the numerical simulation of Eq. (2) and each simulation run was terminated after second iteration step. The following procedure describes the proposed parallel algorithm. In the simulation, Δt was assigned to one.

0. Set t=0.
1. The initial values of $U_{XY}(0)$ for X=1,..,n Y=1,..,n X≠Y are assigned to zero.
2. Evaluate values of $V_{XY}(t)$ based on the binary function, Eq. (1).
3. Use the motion equation Eq.(2) to compute $\Delta U_{XY}(t)$ for X=1,..,n Y=1,..,n X≠Y.

$$\Delta U_{XY}(t) = -g(N_X, N_Y)\left(\sum_{i \neq X}^{n} g(N_i, N_Y)\, V_{Xi} - 1\right) \qquad (4)$$

4. Compute $U_{XY}(t+1)$ based on the first order Euler method:

$$U_{XY}(t+1) = U_{XY}(t) + \Delta U_{XY}(t) \quad \text{for } X=1,..,n \quad Y=1,..,n \quad X \neq Y \qquad (5)$$

5. Increment t by 1.
6. If t>2 then terminate this procedure, else go to step 2.

Fig. 1 shows a simple simulation result using the proposed algorithm to sort 15 numbers which were randomly generated. The simulation was applied to various size of problems up to n=1000. All of the results show that the computation time is constant, namely two-step.

3. Architecture

Fig. 2 depicts the architecture of the proposed parallel sorting system based on Eq. (2) where it is composed of $n \times (n-1)$ processing elements. S_{XY} represents a switch which is controlled by a comparator C_{XY}. Fig. 3 shows the detail circuit diagram of the 12th neuron, that is X=1 and Y=2. S_{12} is the single-pole-double-throw switch which turns on when N_2 is greater equal N_1. This function is to implement $g(N_X, N_Y)$ in Eq. (2). S_{22} thorough S_{n2} are also the switches which operate according to the results of the comparators. Each comparator compares two input values, N_i and N_2 ($i = 2, .. ,n$), and if N_2 is greater equal N_i then switch S_{i2} turn on. The switches and the comparators implement the summation term in Eq. (2). The output of S_{12} is the input to the neuron, that is $\frac{dU_{12}}{dt}$.

Conclusion

The proposed parallel algorithm requires two and only two iteration steps to sort unsorted elements regardless of the problem size. The system based on the proposed algorithm uses $n \times (n-1)$ neurons for n-1 sorting problems.

References

[1] F. T. Leighton, "Tight bounds on the complexity of parallel sorting," Proc. of the ACM sym. on Theory of Computing, pp.71-80, ACM, 1984.

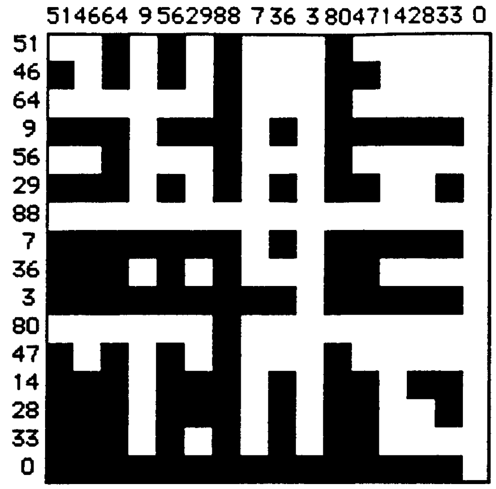
[2] Y. Takefuji, and K. C. Lee, "A near-optimum parallel planarization algorithm," Science, 245, pp.1221-1223, Sept. 1989.

[3] Y. Takefuji, and K. C. Lee, "A parallel algorithm for tiling problems," the IEEE Trans. on Neural Networks, 1, March 1990.

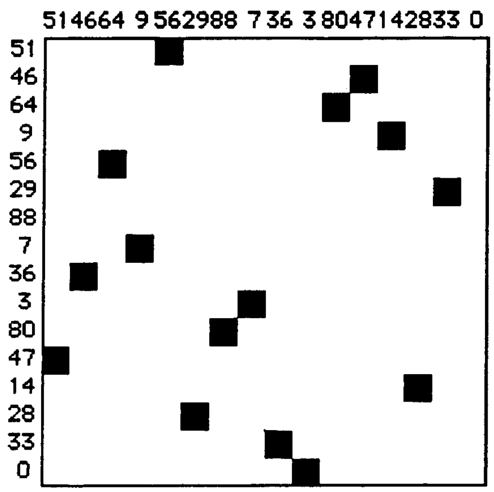
**** sorting *****

n=16

(a) time step =1



(b) time step =2



Sorting result N_{π} = (3 7 9 14 28 29 33 36 46 47 51 56 64 80 88)

Fig. 1 The simulation result of n= 16

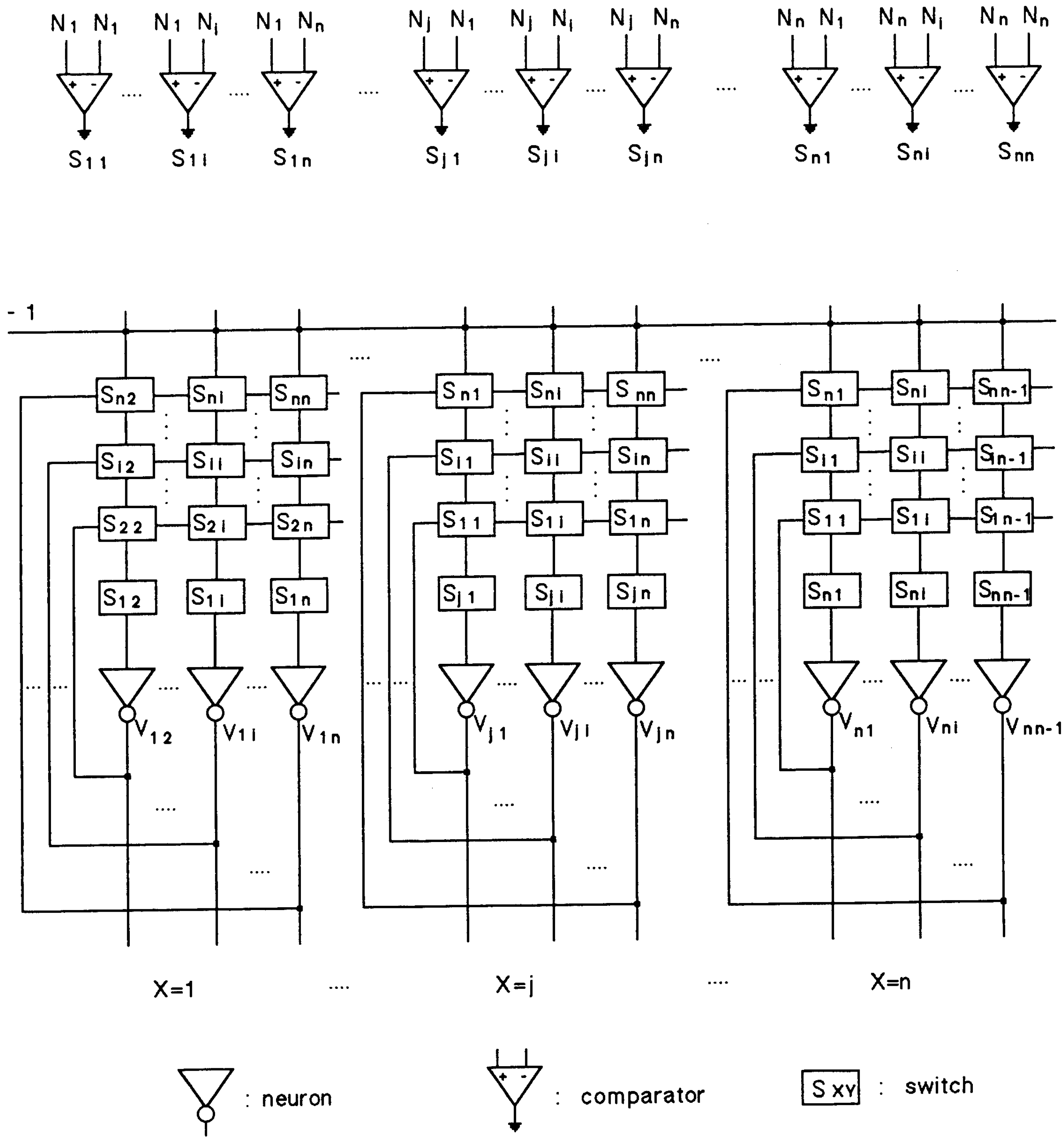


Fig. 2 A circuit diagram of the sorting neural network

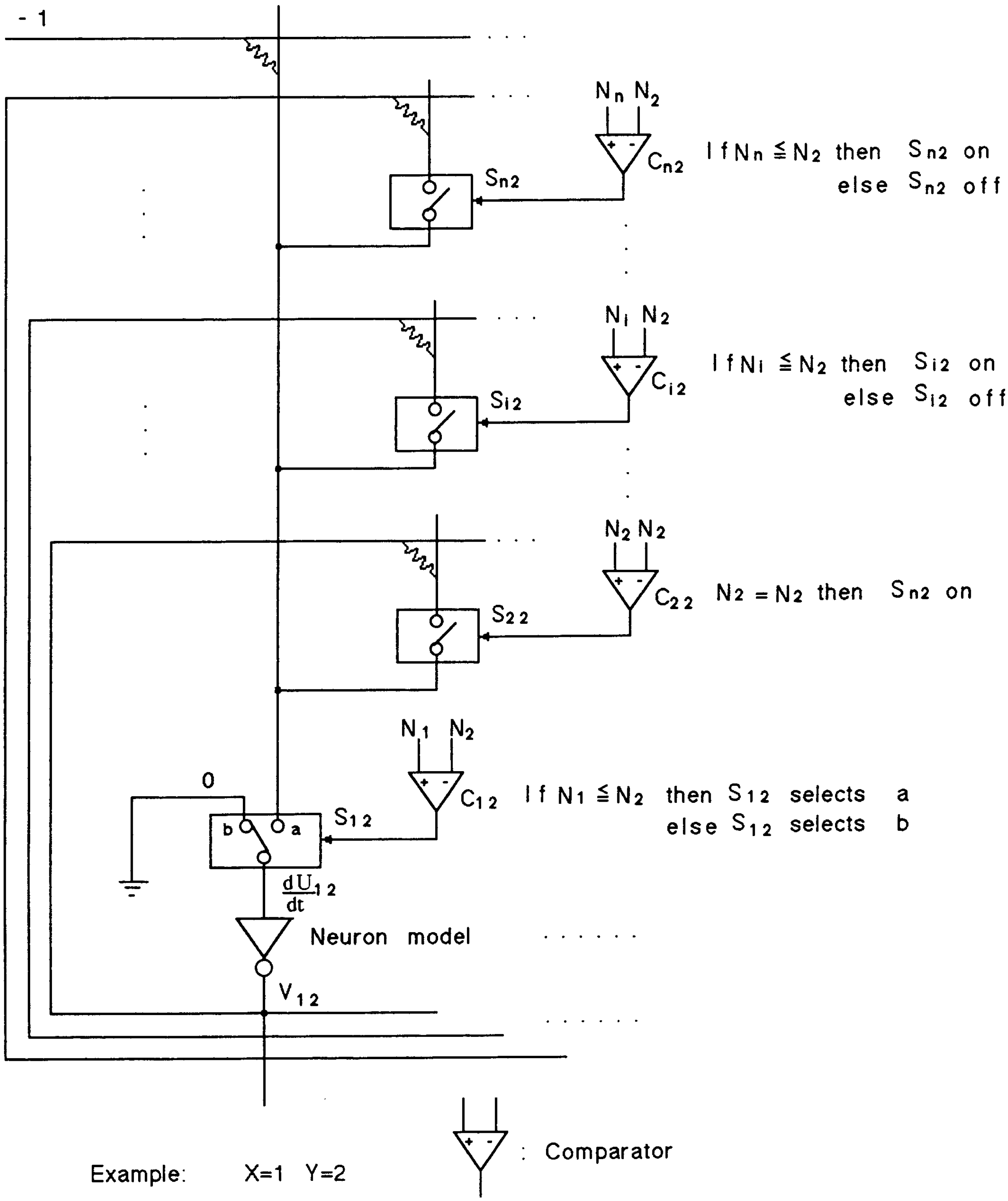


Fig. 3 A detail circuit diagram of one neuron