Notice further that

$$\tilde{K} = T^{-1}KT^{-t} = \begin{bmatrix} 1 & * \\ * & 1 \end{bmatrix}$$

and

$$\tilde{W} = T^{t}WT = \begin{bmatrix} W_{11}K_{11} & * \\ * & W_{22}K_{22} \end{bmatrix} \equiv \begin{bmatrix} \tilde{W}_{11} & * \\ * & \tilde{W}_{22} \end{bmatrix}.$$

Hence

$$\tilde{G} = \sum_{i=1}^{2} \tilde{W}_{ii}\tilde{K}_{ii} = \sum_{i=1}^{2} W_{ii}K_{ii} = G_0 = 0.032\,428.$$

On comparing the above result with that in [2, sect. IV], it is observed that our $\tilde{G}$ is slightly smaller than the optimized unit noise in [2], which is 0.034 951. The small difference may be due to the truncation method used in [2] to obtain matrices $\boldsymbol{K}$ and $\boldsymbol{W}$, which may have resulted in an imperfect realization. The basic problem with the Roesser realization used in [2] is that it is of higher order than necessary, which has resulted in more multipliers and, therefore, more noise.

References

[1] W.-S. Lu and A. Antoniou, "Synthesis of 2-D state-space fixed-point digital-filter structures with minimum roundoff noise," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 965–973, Oct. 1986.

[2] T. Hinamoto, T. Hamanaka, and S. Maekawa, "A generalized study on the synthesis of 2-D state-space digital filters with minimum roundoff noise," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 1037–1042, Aug. 1988.

[3] T. Lin, M. Kawamata, and T. Higuchi, "A unified study on the roundoff noise in 2-D state-space digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 724–730, July 1986.

A Super-Parallel Sorting Algorithm Based on Neural Networks

YOSHIYASU TAKEFUJI AND KUO-CHUN LEE

***Abstract*—A new neural network parallel algorithm for sorting problems is presented in this paper. The proposed algorithm using $O(n^2)$ processors requires two and only two steps, not depending on the size of the problem, while the conventional parallel sorting algorithm using $O(n)$ processors proposed by Leighton needs the computation time $O(\log n)$. A set of simulation results substantiates the proposed algorithm. The hardware system based on the proposed parallel algorithm is also presented in this paper.**

I. Introduction

Sorting is one of the fundamental operations in computer science and engineering. The conventional sequential algorithm based on comparisons of pairs of elements must have complexity: $O(n \log n)$. In 1968, Batcher introduced a parallel sorting algorithm with time complexity $O(\log^2 n)$ using $2^{k-2}k(k+1)$ comparators where $n = 2^k$ are the unsorted elements [1]. Leighton shows the algorithm using n processors in time $O(\log n)$ [2]. Alon and Azar have claimed $\Theta(\log n / \log(1 + p/n))$ algorithms using p processors [3].

Since the advent of VLSI technology, the hardware cost has become negligible. In this paper a new parallel distributed algorithm to sort a list of n unsorted elements is presented. The algorithm using $O(n^2)$ processors requires two and only two iteration steps regardless of the size of the problem.

Processors used in the new algorithm are called neurons (where they perform the function of a simplified biological neuron), or binary neurons. Binary neurons have been successfully used for solving graph planarization problems [4] and tiling problems [5]. The output of the binary neurons is given by

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

where V_i is the output of the ith neuron and U_i is the input to the ith neuron. Interconnection weights, called synaptic weights, between the neurons are determined by the predefined energy function $E(V)$, which describes the penalty quantity where V is an n-dimensional vector: $V = (V_1, V_2, \cdots, V_n)$.

Before discussing the details of our method, first we are going to show how the neural network can perform the parallel gradient descent method. As long as the motion equation of the binary neurons is given by $dU_i/dt = -dE/dV_i$, the predefined energy function E monotonically decreases. The following proofs claim that the state of our neural network is guaranteed to converge to the local minimum under the discrete numerical simulation.

Proofs:

$$\frac{dE}{dt} = \sum_i \frac{dV_i}{dt}\frac{dE}{dV_i}$$

$$= \sum_i \frac{dV_i}{dt}\left(-\frac{dU_i}{dt}\right),$$

where dE/dV_i is replaced by $\left(-\dfrac{dU_i}{dt}\right)$

$$= -\sum_i \left(\frac{dU_i}{dt}\frac{dV_i}{dU_i}\right)\left(\frac{dU_i}{dt}\right)$$

$$= -\sum_i \left(\frac{dV_i}{dU_i}\right)\left(\frac{dU_i}{dt}\right)^2.$$

Let $dV_i(t)/dU_i(t)$ be $(V_i(t+\Delta t) - V_i(t))/(U_i(t+\Delta t) - U_i(t))$. Let $dU_i(t)/dt$ be $(U_i(t+\Delta t) - U_i(t))$. It is necessary and sufficient to consider the following seven cases:

1) $U_i(t+\Delta t) > U_i(t)$, $U_i(t+\Delta t) < 0$, and $U_i(t) < 0$

2) $U_i(t+\Delta t) > U_i(t)$, $U_i(t+\Delta t) \geqslant 0$, and $U_i(t) < 0$

3) $U_i(t+\Delta t) > U_i(t)$, $U_i(t+\Delta t) > 0$, and $U_i(t) \geqslant 0$

4) $U_i(t+\Delta t) < U_i(t)$, $U_i(t+\Delta t) \geqslant 0$, and $U_i(t) > 0$

5) $U_i(t+\Delta t) < U_i(t)$, $U_i(t+\Delta t) < 0$, and $U_i(t) \geqslant 0$

6) $U_i(t+\Delta t) < U_i(t)$, $U_i(t+\Delta t) < 0$, and $U_i(t) < 0$

7) $U_i(t+\Delta t) = U_i(t)$.

If Condition 7) is satisfied, then $dU_i/dt = 0$ must be zero so that $dE/dt = 0$.

Manuscript received April 18, 1989. This work was supported by the National Science Foundation under Grant MIP-8902819. This paper was recommended by Associate Editor M. Ilic.

The authors are with the Department of Electrical Engineering, Center for Automation and Intelligent Systems Research, Case Western Reserve University, Cleveland, OH 44106.

IEEE Log Number 9038550.

If 1), 3), 4), or 6) is satisfied, then dV_i/dU_i must be zero, because of

$$\frac{dV_i}{dU_i}=\frac{V_i(t+\Delta t)-V_i(t)}{U_i(t+\Delta t)-U_i(t)}=\frac{0}{\text{nonzero number}},\quad \text{so that } dE/dt=0.$$

If 2) is satisfied, then dV_i/dU_i must be positive, because of

$$\frac{dV_i}{dU_i}=\frac{V_i(t+\Delta t)-V_i(t)}{U_i(t+\Delta t)-U_i(t)}=\frac{1}{\text{positive number}},\quad \text{so that } dE/dt<0.$$

If 5) is satisfied, then dV_i/dU_i must be also positive, because of

$$\frac{dV_i}{dU_i}=\frac{V_i(t+\Delta t)-V_i(t)}{U_i(t+\Delta t)-U_i(t)}=\frac{-1}{\text{negative number}},\quad \text{so that } dE/dt<0.$$

We can conclude that the energy function E monotonically decreases as long as the motion equation of the binary neurons is given by $dU_i/dt=-dE/dV_i$. Q.E.D.

II. Sorting Neural Networks

$N-1$ positive integers, $N_1, N_2, \cdots, N_{n-1}$, and 0 (zero)—which is a dummy number—are given where the subscript i indicates the location of the register i that contains the number N_i. The goal of sorting is to find a permutation $(\pi_1, \pi_2, \cdots, \pi_{n-1})$ such that $0<N_{\pi_1}<N_{\pi_2}<\cdots<N_{\pi_{n-1}}$. In our new sorting, an $n\times n$ neural network array is provided where each row and column corresponds to the location of a register and the position of permuted order, respectively. The $n\times n$ array actually represents the directed adjacency matrix where one and only one neuron in the ith row $(i=1,\cdots,n)$ will be fired in order to determine the sorting order between N_{π_i} and $N_{\pi_{i+1}}$. 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 predefined energy function based on our neural representation is given by

$$E=\sum_{X=1}^{n}\sum_{Y\neq X}^{n}\frac{A}{(N_Y-N_X)}\sum_{i\neq X}^{n}f(N_Y,N_X)V_{Xi}V_{XY}-\sum_{X=1}^{n}\sum_{Y\neq X}^{n}\frac{A}{(N_Y-N_X)}V_{XY}. \quad (2)$$

Instead of explaining (2), it is much easier to understand the behavior of the neural network using the motion equation of the neuron. The motion equation of the neuron in the Yth column and in the Xth row is derived from the necessary condition of the introduced parallel gradient descent method: $dU_{XY}/dt=-dE/dV_{XY}$. Therefore, from (2), it is given by

$$\frac{dU_{XY}}{dt}=-\frac{A}{N_Y-N_X}\left(\sum_{i\neq X}^{n}f(N_Y,N_i)V_{Xi}-1\right) \quad (3)$$

where

$$f(L,R)=\begin{cases}0, & \text{if } L\leqslant R\\ 1, & \text{otherwise.}\end{cases} \quad (4)$$

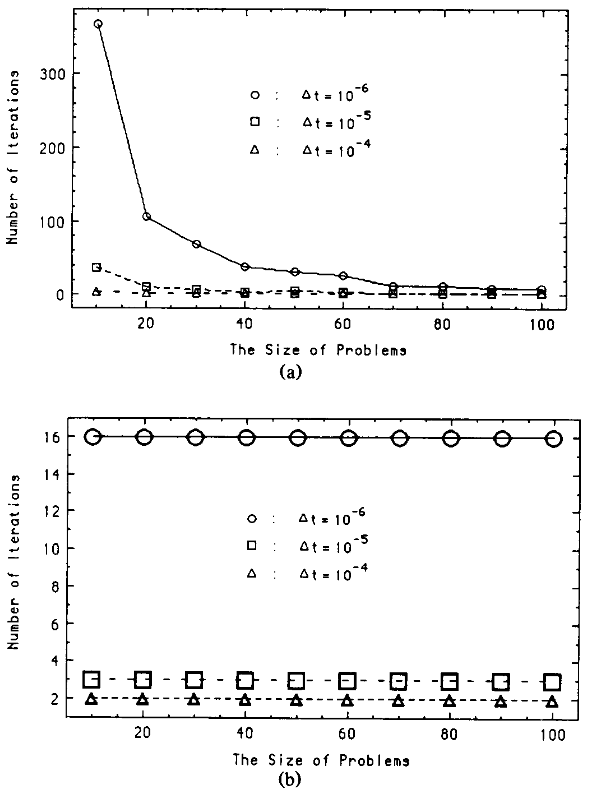


Fig. 1. The relationship between the number of iterations and the problem size. (a) Based on (3). (b) Based on (7).

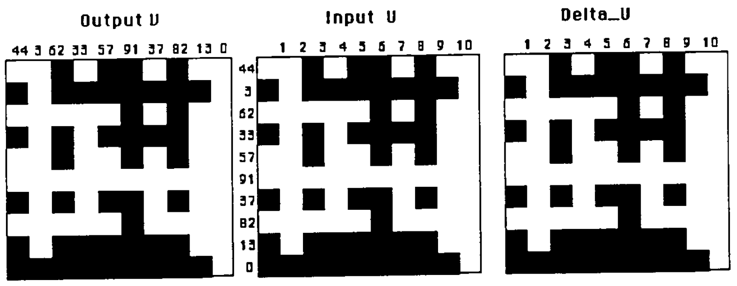
It is very important that all of the initial values at $t=0$ for U_{ij} $(i=1,\cdots,n)$, $(j=1,\cdots,n)$ are set to small negative numbers. In (3), the term $(\Sigma f(N_Y,N_i)V_{Xi}-1)$ forces one and only one neuron to fire per row. When N_Y is greater than N_X, (3) at $t=0$ becomes positive because V_{ij} $(i=1,\cdots,n)$, $(j=1,\cdots,n)$ are zeros. The function of $f(N_Y,N_i)$ plays a key role in our method to determine which permutation connection should remain and which should be removed. In other words, only if N_Y is greater than N_X and the smallest number among N_i $(i=1,\cdots,n-1)$, the term $\Sigma f(N_Y,N_i)V_{Xi}$ is zero so that (3) will be still positive; otherwise it will be negative.

We have already proved that the state of the system is guaranteed to converge to the local minimum. The $A/(N_Y-N_X)$ term and the function $f(N_Y,N_i)$ in (3) will guarantee the state of the system to converge to the global minimum. In order to achieve the global minimum convergence, remember all of the initial values for the inputs U_{ij} $(i=1,\cdots,n)$, $(j=1,\cdots,n)$ must be set to small negative numbers.

Proofs:

Consider the neuron in the Yth column and in the Xth row. The initial values of U_{ij} $(i=1,\cdots,n)$, $(j=1,\cdots,n)$ are negative at $t=0$ so that all of the V_{ij} are zeros, because of the function of binary neurons in (1). If all of the V_{ij} are zeros, then (3) must be positive as long as the value of (N_Y-N_X) is positive. If the value of (N_Y-N_X) is negative, (3) will be negative so that the neuron is not fired. We can conclude that at $t=0$ the negative initial values of U_{ij} $(i=1,\cdots,n)$, $(j=1,\cdots,n)$ forces the neurons whose number is greater than N_X to fire.

```
***** Sorting Problem *****
  Iteration =        1   th
  Delta_max = 0.00100
  U_max  =  0.00099
  Time_step = 0.00100
```



```
***** Sorting Problem *****
  Iteration =        2   th
  Delta_max = 0.00800
  U_max  =  0.00701
  Time_step = 0.00100
```

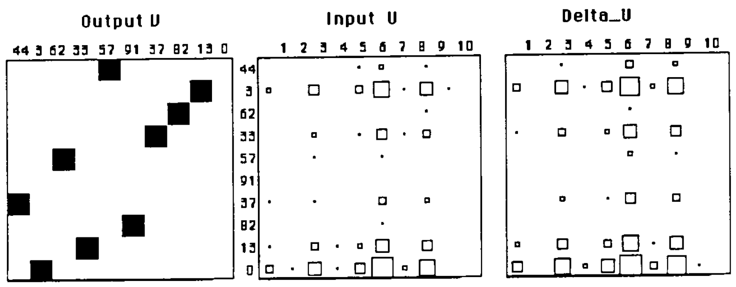


Fig. 2. The convergence of a 10 × 10 sorting neural network to a solution.

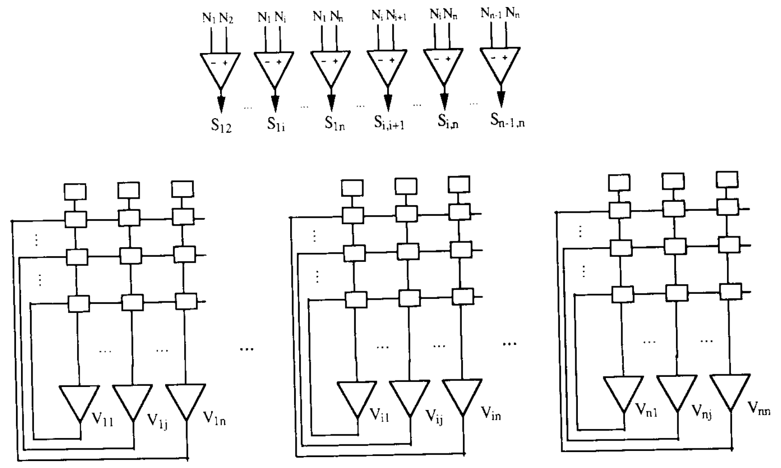


Fig. 3. A parallel sorting neural network.

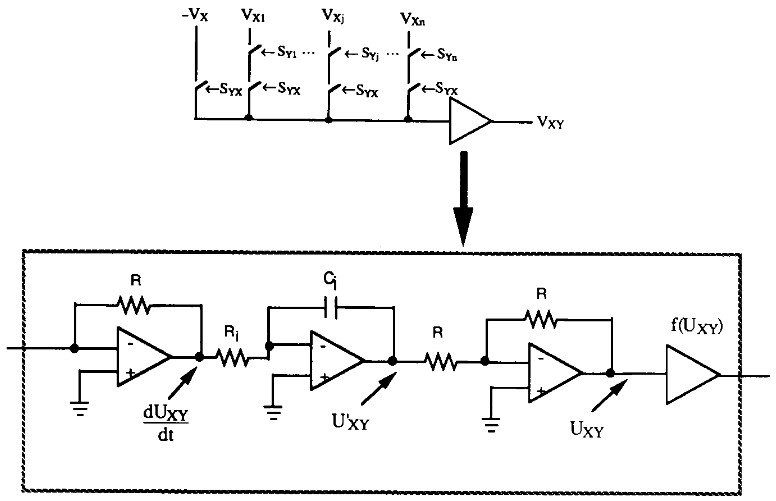


Fig. 4. A circuit diagram of the neuron XY.

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, because the product of $f(N_Y, N_i)$ and V_{Xi} $(i = 1,\cdots,n)$ in (3) enables the neuron in the Yth column and in the Xth row to unfire as long as $N_Y \geqslant N_i$ is satisfied. If the N_Y is the smallest number, then the term $\Sigma f(N_Y, N_i)V_{Xi}$ will be zero so that the neuron is forced to remain fired. Consequently, our method only requires two iteration steps to converge to the global minimum as long as the appropriate number is assigned to Δt, where it must be greater than the absolute values of the initial U_{ij} $(i = 1,\cdots,n)$, $(j = 1,\cdots,n)$ at time $t = 0$.

Q.E.D.

Our numerical simulations of (3) were based on the first-order Euler method; $U_{XY}(t+\Delta t) = U_{XY}(t) + \Delta U_{XY}\Delta t$, where ΔU_{XY} is dU_{XY}/dt. Fig. 1(a) shows the relationship between the size of the problems and the number of iterations for the global minimum convergence where Δt was varied 10^{-4} to 10^{-6} and the initial values of U_{XY} were set to 1/70000. Unsorted positive numbers were generated randomly. When the absolute values of the initial U_{ij} $(i = 1,\cdots,n)$ $(j = 1,\cdots,n)$ are greater than Δt, more than two iterations are required. When Δt is greater than those, the state of the system always converges to the global minimum with two and only two iteration steps.

The energy function in (2) follows the quadratic form:

$$E = \sum_{X=1}^{n}\sum_{Y=1}^{n}\sum_{i=1}^{n}\sum_{j=1}^{n} T_{XY,ij}V_{XY}V_{ij} + \sum_{X=1}^{n}\sum_{Y=1}^{n} V_{XY}I_{XY}. \quad (5)$$

Therefore, the conductance matrix $T_{XY,ij}$ is given by

$$T_{XY,ij} = -\frac{A}{N_Y - N_X}(1-\delta_{XY})\delta_{Xi}f(N_Y,N_j)(1-\delta_{Xj}) \quad (6)$$

where δ_{pq} is 1 if $p = q$, and 0 otherwise.

In order to remove the expensive term $A/(N_Y - N_X)$ from (6), which must use the variable conductance devices, the following motion equation is created:

$$\frac{dU_{XY}}{dt} = -f(N_Y,N_X)\left(\sum_{j\neq X}^{n} f(N_Y,N_j)V_{Xj} - 1\right). \quad (7)$$

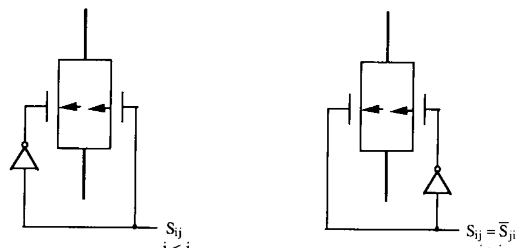


Fig. 5. A comparator signal and an analog switch.

Hence the new conductance matrix is given by

$$T_{XY,ij} = -f(N_Y,N_X)(1-\delta_{XY})\delta_{Xi}\cdot f(N_Y,N_j)(1-\delta_{Xj}). \quad (8)$$

It is very important to notice that (8) does not require the variable conductance devices. Fig. 1(b) shows the relationship between the number of iterations and the size of problems based on (8). The number of iteration steps can be given by the following formula:

$$\text{The number of iteration steps} = \text{Integer}\left[\frac{|U_{ij}(0)|}{\Delta t}\right] + 2 \quad (9)$$

where $U_{ij}(0)$ is the initial value at time $t = 0$ $(i = 1,\cdots,n$ and $j = 1,\cdots,n)$.

The sorting neural network based on (8) needs n^2 binary neurons and $n(n-1)/2$ comparators to sort $n-1$ unsorted elements. Fig. 2 shows a simple example to sort nine numbers (44,3,62,33,57,91,37,82,13) using the 10×10 neural network array. The simulation result indicates $0 < 3 < 13 < 33 < 37 < 44 < 57 < 62 < 82 < 91$.

We have investigated the problem size up to $n = 1000$. All of our results showed that the computation time is constant, namely, two-step, when the following condition is satisfied:

$$U_{ij}(0) < \Delta t, \qquad i = 1,\cdots,n;\ j = 1,\cdots,n.$$

Fig. 3 depicts the architecture of the proposed parallel sorting system based on (8) where it is composed of n^2 neurons and $n(n-1)/2$ comparators. Fig. 4 shows the details of the neuron XY where the circuit is not yet minimized for the actual implementation. In Fig. 4, the operational amplifier on the first stage sums all of the inputs, which satisfies the following equation:

$$\frac{dU_{XY}}{dt} = -R\left(\sum_{j=1}^{n} f(N_Y, N_X) f(N_Y, N_j)\frac{V_{Xj}}{2R_{\text{on}}} - \frac{V_X}{R_{\text{on}}}\right) \quad (10)$$

where R_{on} is the on-resistance when a single analog switch is on.

The second operational amplifier performs integration of dU_{XY}/dt, which is given by the following equation:

$$U'_{XY} = -\frac{1}{R_i C_i}\int\left(\frac{dU_{XY}}{dt}\right)dt. \quad (11)$$

The third operational amplifier generates U_{XY} where $U_{XY} = -U'_{XY}$. In the last stage, $f(U_{XY})$ follows the nondecreasing function.

Fig. 5 shows how to control the signal from the comparator to the analog switch. Remember that $n(n-1)/2$ comparators are required for sorting $n-1$ numbers.

III. 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^2 neurons and $n(n-1)/2$ comparators for $n-1$ sorting problems.

References

[1] K. E. Batcher, "Sorting networks and their applications," in *Proc. of the Spring Joint Computer Conf.*, vol. 32, AFIPS Press, 1968.
[2] F. T. Leighton, "Tight bounds on the complexity of parallel sorting," in *Proc. of the ACM Symp. on Theory of Computing*, pp. 71–80, 1984.
[3] N. Alon and Y. Azar, "The average complexity of deterministic and randomized parallel comparison-sorting algorithms," *SIAM J. Computing*, vol. 17, Dec. 1988.
[4] Y. Takefuji and K. C. Lee, "A near-optimum parallel planarization algorithm," *Science*, 245, pp. 1221–1223, Sept. 1989.
[5] ____, "A parallel algorithm for tiling problems," *IEEE Trans. Neural Networks*, vol. 1, pp. 143–145, Mar. 1990.

On the Explicit Formulas for the Elements in Low-Pass Ladder Filters

PAULO ANTONIO MARIOTTO

***Abstract*—As a contribution in searching the explicit formulas for the elliptic-function filter, this paper develops formulas by which the elements of a resistively terminated, linear low-pass *LC* ladder filter can be expressed in terms of the filter's natural frequencies as well as its loss poles and zeros. After a general approach to the problem, a special case illustrates the formulas.**

Manuscript received November 10, 1987; revised May 2, 1989. This paper was recommended by Associate Editor T. R. Viswanathan.
The author is with the Department of Electrical Engineering, Escola Politécnica da Universidade de São Paulo, CEP. 05508, São Paulo, Brazil.
IEEE Log Number 9038551.

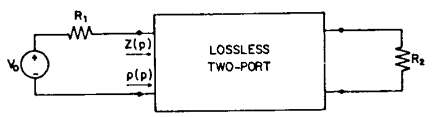


Fig. 1. Lossless network inserted between source and load.

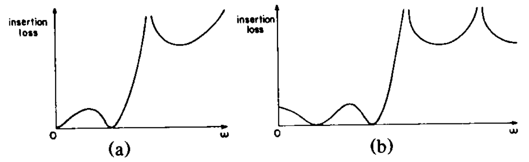


Fig. 2. Insertion loss of some low-pass filters.

I. Introduction

The modern filter synthesis, given a prescribed insertion-loss between a resistive source and a resistive load, is a classical procedure (which started with Darlington's work [1]) presented in many texts on network synthesis [2]. The method consists of, given the insertion-loss function, determining the squared-magnitude $|\rho(j\omega)|^2$ of the reflection coefficient $\rho(p)$, then getting a stable $\rho(p)$, and, finally, deriving the corresponding driving-point impedance $Z(p)$. From this $Z(p)$, a lossless network terminated by a resistance can be found, satisfying the prescribed insertion-loss.

For some special data, the resulting $Z(p)$ can be developed in continued-fraction expansion, thus yielding a network in ladder form. Some work in the past obtained explicit formulas for the elements in ladder form for some configurations of the poles and zeros of $\rho(p)$ but under the assumption of "all points of infinite loss at infinity" [3]–[5]. Orchard [6] has given explicit formulas for the elements allowing finite frequencies of infinite loss but starting with the driving-point impedance of the unterminated lossless filter. In the present paper, a compromise is achieved between these two approaches.

By using the so-called Newton's formulas [7], which relate the coefficients and roots of polynomials, it is shown how to express Orchard's formulas as functions of the poles and zeros of $\rho(p)$, as well as the frequencies of infinite loss. Further, as the formulas are expressed by determinants, it is shown how to linearly combine their columns so as to get more compact determinants. A numerical example follows the mathematical derivation.

II. Preliminaries

Let

$$\rho(p) = G(p)/H(p) \quad (1)$$

be the reflection coefficient at the input of a resistively terminated lossless network, as illustrated in Fig. 1, where

$$H(p) = h_0 p^n + h_1 p^{n-1} + h_2 p^{n-2} + \cdots + h_n \quad (2)$$

is a Hurwitz polynomial on the complex frequency variable p and where

$$G(p) = g_0 p^n + g_1 p^{n-1} + \cdots + g_n \quad (3)$$

with $0 < g_0 \leqslant h_0$ and $|g_n| < h_n$. It is also assumed that $|\rho(j\omega)| \leqslant 1$ for all ω.