Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

POLYNOMIAL EVALUATION AND INTERPOLATION IN VLSI

B.CODENOTTI, G.LOTTI, L.MATROPIETRO

Nota interna B 4-14

Maggio 1987

POLYNOMIAL EVALUATION AND INTERPOLATION IN VLSI

B. CODENOTTI[1], G.LOTTI[2] and L.MASTROPIETRO[3]

[1]Istituto di Elaborazione dell'Informazione - CNR, via S.Maria 46, PISA, ITALY.

[2]Dipartimento di Scienze dell'Informazione - University of Pisa, Corso Italia 40, PISA, ITALY.

[3]Geomath, Via Cavour 43, PISA,ITALY.

ABSTRACT

The area-tima complexity of polynomial interpolation, and of the evaluation of a polynomial at given points is studied. We will show VLSI designs producing bounds close to the lower bounds also derived.

KEY WORDS. Area-Time Complexity, VLSI, Interpolation, Polynomial Evaluation.

1. INTRODUCTION

In this paper, the problem of finding a polynomial

$$P(x) = \sum_{i=0}^{n} a_i x^i$$ of degree less or equal than n,

such that $P(x_i) = f_i$, where $\{ (x_i, f_i), i=0,1,\ldots,n \}$ is a set of given points, i.e. the polynomial interpolation problem, is studied.

The coefficients of the interpolating polynomial are the solution of the linear system $V\underline{a} = \underline{f}$, where V is the $(n+1)\times(n+1)$ Vandermonde matrix $V = (v_{ij})$, $v_{ij} = x_{i-1}^{j-1}$, and $\underline{f} = [\, f_1 \; f_2 \; \ldots \; f_n \,]^T$.

It comes out that the interpolation problem is equivalent to the solution of a particular linear system, with Vandermonde coefficient matrix.

In the next section, the question of lower bounds is addressed.

In section 3, a VLSI design for the computation of the value of the interpolating polynomial at a given point with $AT^2 = O(n^4)$ is obtained; another VLSI network producing the same upper bound, is shown for the computation of the coefficients of the required polynomial.

Moreover, the VLSI implementation of Estrin algorithm [3] for the evaluation of a given polynomial at one point is

described, with $AT^2 = O(n \log^3 n)$.

The evaluation of a given polynomial at n points is also considered; this problem is equivalent to the computation of the product of a Vandermonde matrix by a vector.

A VLSI design derived from Estrin algorithm yields the AT^2 upper bound $O(n^2 \log^3 n)$, while a VLSI design for generic matrix-vector product, e.g. mesh of trees, produces $AT^2 = O(n^2 \log^4 n)$ [4].

2. Area-Time lower bounds

In this section we will prove a lower bound to the problem of evaluating a polynomial of degree less or equal than n at a given point, with respect to the areax(time)^2 measure.

Let a(i), i=1,...,n be the coefficients of the given polynomial. As usual in numerical analysis, we will assume that a fixed number of digits are used to represent such coefficients. The evaluation of the polynomial is not easier than the computation of the sum of the coefficients a(i), i=1,...,n.

For the latter problem it is possible to prove the lower bound $AT^2=\Omega(n\log^2 n)$.

Indeed, we will use the fact that the lower bound $AT^2=\Omega(n\log^2 n)$ holds for the "counter" problem, i.e. the problem of counting the number of occurrences of 1-s in a binary string of lenght n. This result easily follows by applying the technique used by Johnson for integer addition. Such technique is based on the input-output functional dependence, which holds

for the counter problem as well.

In the special case of one digit used to represent the coefficients of the polynomial, we have that the problem of computing the sum of such coefficients is equivalent to the counter problem. Therefore the lower bound $AT^2=\Omega(n\log^2 n)$ holds for polynomial evaluation, as well.

In the next section, we will show a design which produces an upper bound to AT^2, optimal up to the exponent of the logarithm.

For what concerns the interpolation problem, it is equivalent to the solution of a special linear system, with Vandermonde coefficient matrix. The upper bound derived in the next section attains the lower bound holding for general linear systems solution, and it is characterized by a simple and modular structure.

3. VLSI designs for polynomial interpolation and evaluation

Let $\overline{\prod}_n$ denote the set of complex or real polynomials of degree less or equal to n.

Given n+1 arbitrary points (x_i, f_i), $i = 0,1,\ldots,n$, the problem of finding the polynomial $P(x) \in \overline{\prod}_n$ such that

$$P(x_i) = f_i, \quad i=0,1,\ldots,n,$$

i.e. the interpolating polynomial, is considered.

In this section, two well known algorithms for the computation of the interpolating polynomial are taken into account, namely Neville algorithm [5], which is particularly suitable to find the value of the interpolating polynomial at a given point, and Newton formula [5], which can be efficiently used to compute the coefficients of the required polynomial.

The basic step of Neville algorithm is described by the following recurrent formula:

$$P_i(x) = f_i ,$$

$$P_{i_0 i_1 \ldots i_k}(x) = \frac{(x - x_{i_0}) P_{i_1 \ldots i_k}(x) - (x - x_{i_k}) P_{i_0 \ldots i_{k-1}}(x)}{x_{i_k} - x_{i_0}}, \qquad (2.1)$$

where $P_{0\,1\,\ldots\,n}(x)$ is the required value of the polynomial.

In fig. 1, a VLSI design is presented, in the case n = 3; processor $[i_0\ i_1\ \ldots\ i_k]$ receives as input the values $P_{i_1 \ldots i_k}(x)$ and $P_{i_0 \ldots i_{k-1}}(x)$, computed respectively by processors $[i_1 \ldots i_k]$ and $[i_0 \ldots i_{k-1}]$, together with the values x, x_{i_0}, x_{i_k} and computes the value $P_{i_0 \ldots i_k}(x)$, according to formula (2.1).

Observe that processors denoted by dashed squares

in fig.1, send x_0 (x_n) to the adjacent processor, while each other processor $[i_0 \ldots i_k]$ send x_{i_k} (x_{i_0}) to processor $[i_{-1} i_0 \ldots i_k]$ ($[i_0 \ldots i_k \; i_{k+1}]$).

At the end of the computation, processor $[\,0\;1 \ldots n\,]$, holds the result.

If each processor is supposed to have two storage cells, the values $x_0, \ldots, x_n$ can be sent to the proper processor before starting the computation.

Neville algorithm is not practical to compute the coefficients of the interpolating polynomial. For this purpose, it is convenient to use Newton formula, that is

$$\begin{cases} f(x_i) = f_i \,, \\ f(x_i, x_{i+1}, \ldots, x_{i+k}) = \dfrac{f(x_{i+1}, \ldots, x_{i+k}) - f(x_i, \ldots, x_{i+k-1})}{x_{i+k} - x_i} \,. \end{cases}$$

The required coefficient a_k, corresponding to the k-th power of x, results to be $a_k = f(x_0, x_1, \ldots, x_k)$, $k = 0, 1, \ldots, n$.

The design for this algorithm is very similar to that one shown in fig.1, a part from the different computation performed by each processor. Moreover, the required coefficients are available, at different stages of the computation, in processors $[0]$, $[0\;1]$, $\ldots$, $[0\;1 \ldots n]$.

In both cases, it is convenient to rearrange the networks, in order to obtain the upper diagonal part of an nxn mesh of processors (see fig.2). It readily follows that

$A = O(n^2)$ and $T = O(n)$.

Note that the computation proceeds by diagonals; therefore the circuit is very suitable for a pipelined implementation.

The output period P of the network, defined as the maximum time between two successive data passages at any output port when the circuit is used in a pipelined fashion, is $P = O(1)$; then $AP^2 = O(n^2)$ and, moreover, the complexity $AT^2 = O(n^4)$ is obtained for the computation either of the interpolating polynomial at n given points or of n interpolating polynomials at one point.

REMARK

Note that if $x_i = w^i$, $i = 1,2,\ldots,n$, w being a n-th principal root of the unity, then the Vandermonde matrix associated to the interpolation problem becomes the Fourier matrix [2]; for this particular interpolation problem (trigonometric interpolation) the well known optimal VLSI designs for DFT [1] can be used.

We turn now to the problem of evaluating a given polynomial at given points.

Let $a_0, a_1, \ldots, a_n$ be the coefficients of a polinomial

$P \in \overline{\Pi}_n$.

The following problems are considered:

3.1 evaluate P at a given point x,

3.2 evaluate P at n different points.

Note that problem 3.1 is not easier than evaluating the expression

$$\sum_{i=0}^{n} a_i ,$$

for which the ~~trivial~~ lower bound $AT^2 = \Omega(n \log^2 n)$ ~~can~~ has been proved, in section 2.

In the following, some VLSI designs are derived, which are optimal up to logarithmic factors.

Problem 3.1 can be solved by Estrin algorithm [3], which computes P(x) as

$$P(x) = Q(x)\, x^{n/2+1} + R(x). \tag{3.3}$$

Let $n = 2^k$; then a recursive VLSI design can be readily obtained from (3.3) (see fig.3).

Let us consider the network performing the n-th power of x, in time $T_1(n) = O(\log n)$ and area $A_1(n) = L_1(n)\, H_1(n)$, where

the width $L_1(n)$ and the height $H_1(n)$ can be chosen respectively of order $O(\log n)$ and $O(1)$.

Then, the complexity analysis of the network showed in fig.3, produces the following bounds:

$$T(n) = \max \{ T_1(n/2), T(n/2) \} = O(\log n),$$

$$L(n) = \max \{ L_1(n/2), L(n/2) \} + O(1) = O(\log n),$$

$$H(n) = 2\,H(n/2) + H_1(n/2) = O(n).$$

Hence, the area$\times$(time)^2 complexity is

$$O(n^3 \log n).$$

Problem (3.1) can also be solved by Dorn algorithm [3], which computes

$$\begin{cases} q_0^k(x) = a_0^k + a_k^k x + \dots + a_{(k-1)k}^{(k-1)k} x \\ q_1^k(x) = a_1^k + a_{k+1}^k x + \dots + a_{(k-1)k+1}^{(k-1)k} x \\ \quad \vdots \\ q_{k-1}^k(x) = a_{k-1}^k + a_{2k-1}^k x + \dots + a_n^{(k-1)k} x^{\,}, \end{cases}$$

and then

$$P(x) = q_0(x^k) + x\,q_1(x^k) + \ldots + x^{k-1}\,q_{k-1}(x^k).$$

Let $n=k^2$; then a straightforward VLSI implementation of this algorithm can be obtained according to the following three stages:

0. Compute $x, x^2, \ldots, x^{k-1}$ and $x^k, x^{2k}, \ldots, x^{(k-1)k}$,

1. Compute the matrix-vector product

$$\begin{pmatrix} a_0 & a_k & \ldots & a_{(k-1)k} \\ a_1 & a_{k+1} & \ldots & a_{(k-1)k+1} \\ \cdot & \cdot & \ldots & \cdot \\ a_{k-1} & a_{2k-1} & \ldots & a_n \end{pmatrix} \begin{pmatrix} 1 \\ x^k \\ \cdot \\ \cdot \\ x^{(k-1)k} \end{pmatrix},$$

2. Compute $P(x)$ as a scalar product.

The area-time complexity of this design would be of the same order of the previous one up to logarithmic factors, if the input and the output nodes of the module performing matrix-vector product are not required to lie on the border of the circuit.

Let us consider the evaluation of a given polynomial at n points; it is easy to see that this problem is equivalent to the computation of the product of a Vandermonde matrix by a

vector.

For this problem, the lower bound $\Omega(n^2)$ holds, since the Fourier matrix, for which this lower bound has been proved [6], is a special Vandermonde matrix.

A VLSI design for generic matrix-vector product, using the structure of mesh of trees [4], produces $AT^2 = O(n^2 \log^4 n)$ for this problem.

REFERENCES

[1] G. Bilardi and M. Sarrafzadeh, Optimal Discrete Fourier Transform in VLSI, International Workshop on Parallel Computing and VLSI, Amalfi, 1984.

[2] P.Davis, Circulant Matrices, Wiley Interscience, New Jork, 1979.

[3] D.E. Knuth, The Art of Computer Programming, Vol.2, Seminumerical algorithms, Addison-Wesley, 1969.

[4] F.I.Leighton, New Lower Bound Tecniques for VLSI, Proceedings of the 22nd Annual IEEE Symposium on Foundations of Computer Science, 1981, pp.1-12.

[5] J. Stoer, Einfuhrung in die Numerische Mathematik, alysis Springer Verlag, Berlin, 1973.

[6] C.D. Thompson, Area-Time Complexity for VLSI, Proc. 11th Annual ACM Symp. on the Theory of Computing (SIGACT), 1979, pp.81-88.

FIGURE LEGENDS:

Fig.1 Module structure.

Fig.2 Mesh of PEs.

Fig.3. Recursive VLSI design.

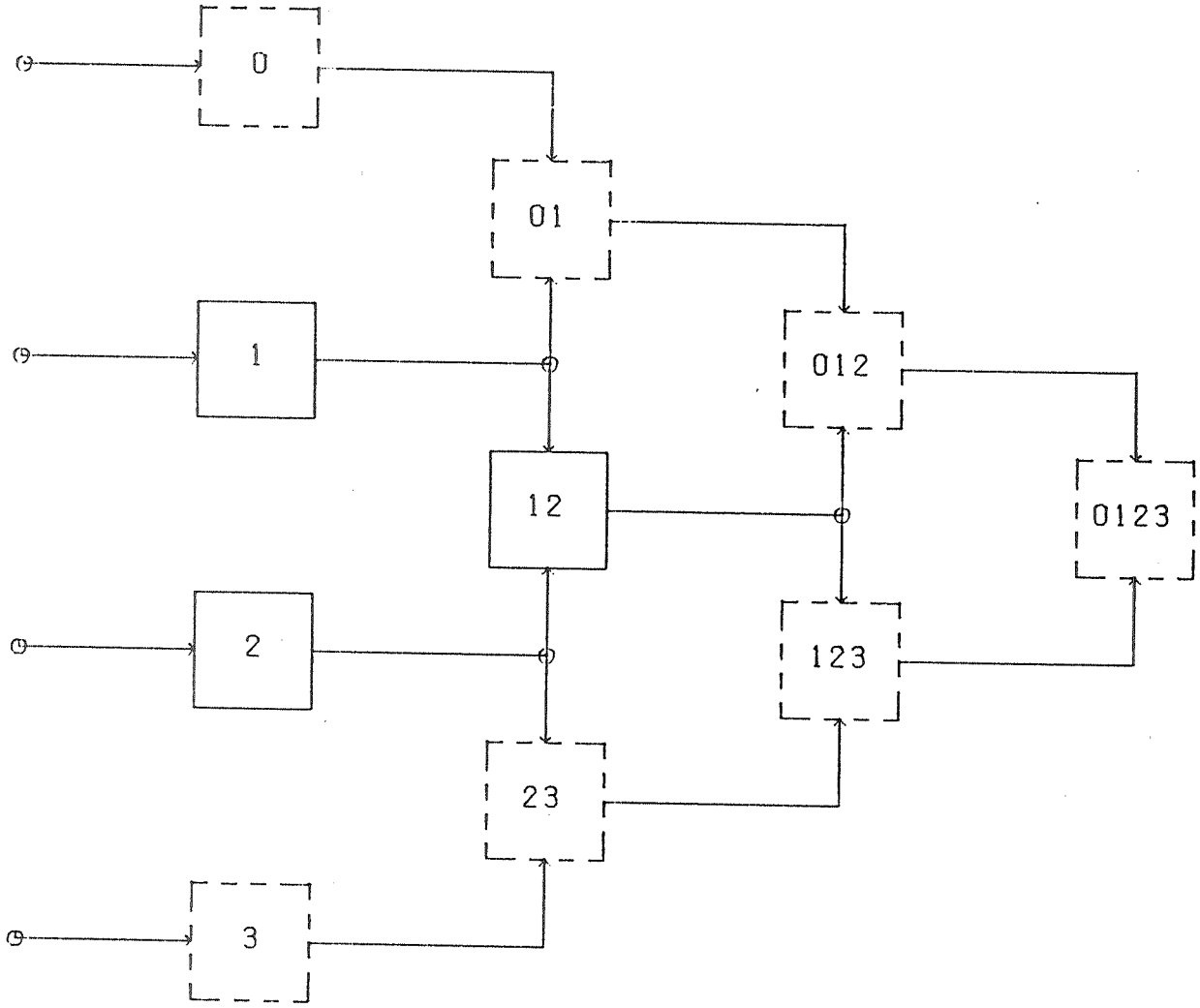
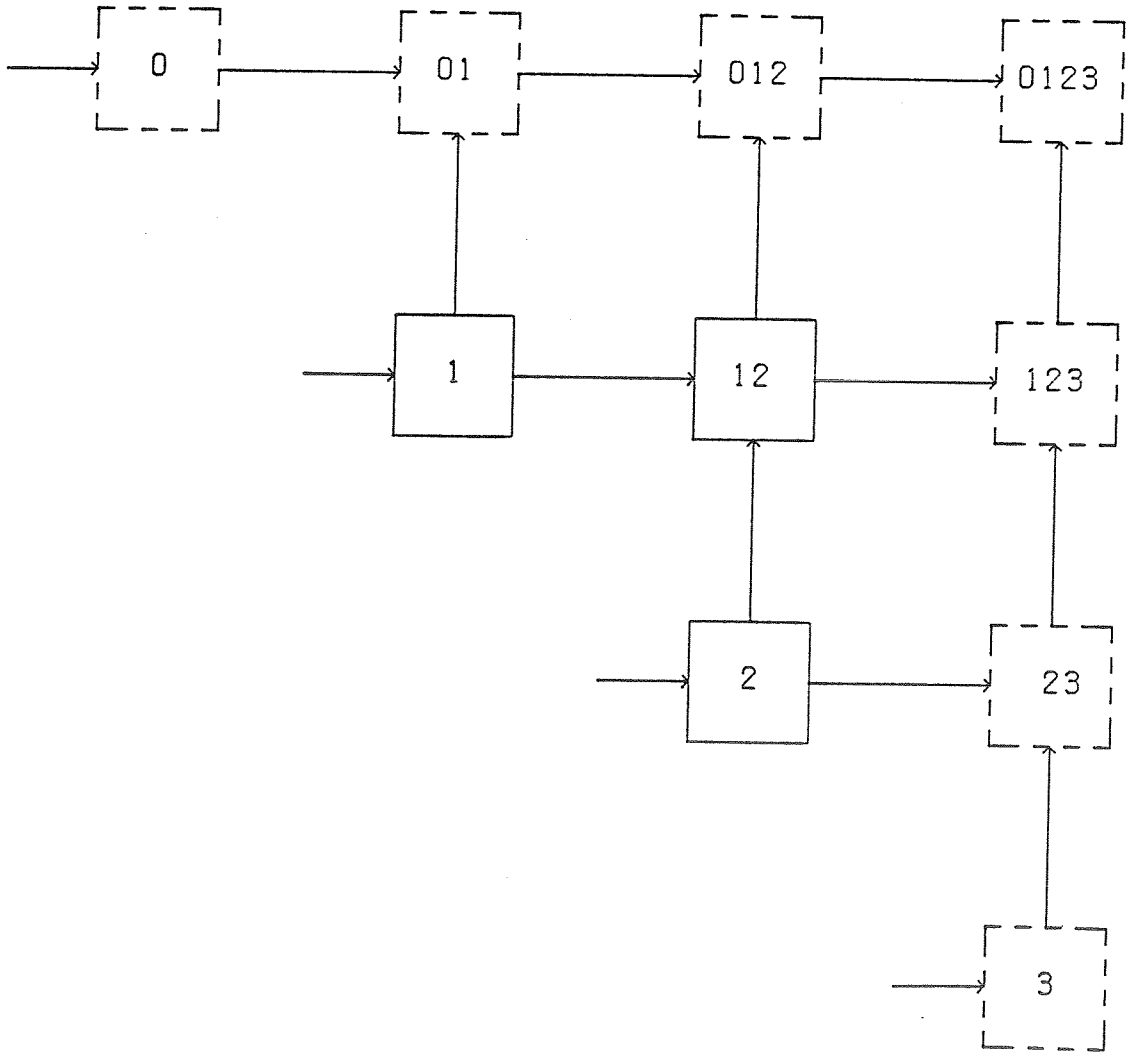


Fig. 1 Module structure.

FIG. 2



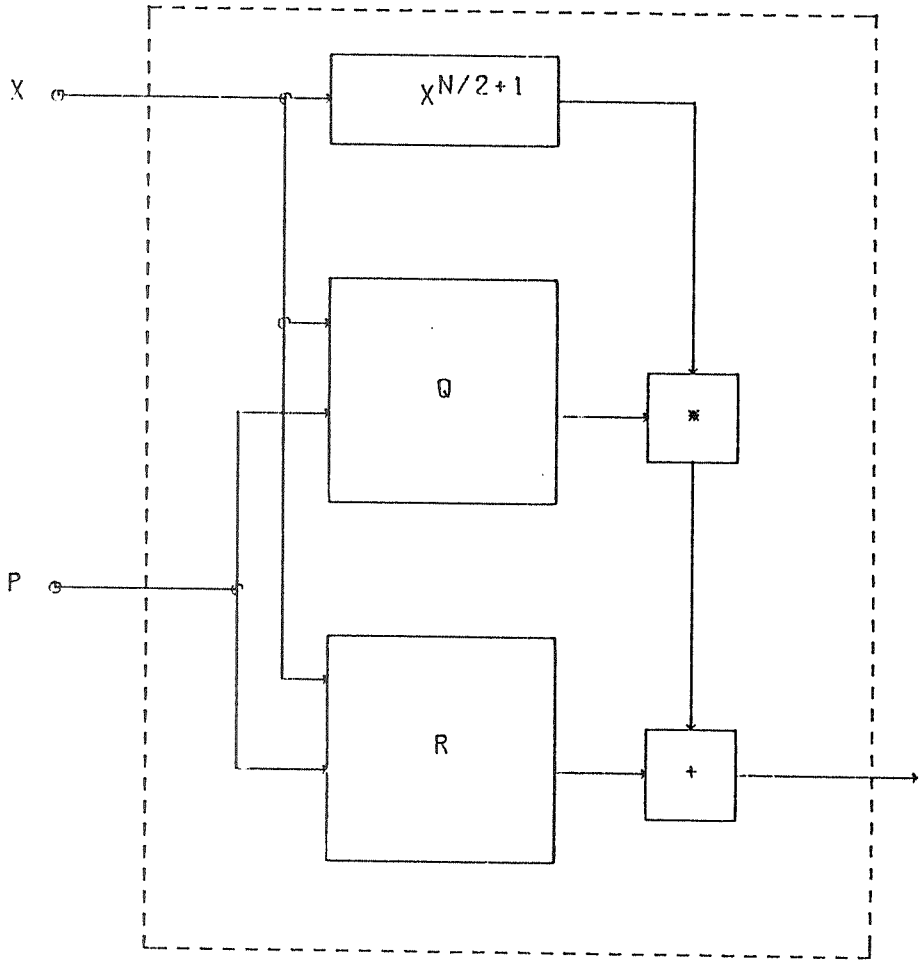


Fig. 3. Recursive VLSI design.