VLSI IMPLEMENTATION OF THE CAPACITANCE MATRIX METHOD

P. Favati, G. Lotti, F. Romani

Nota interna B04-01

Gennaio 1987
1. Introduction

Often, in numerical analysis, a linear system $Ax = b$ has to be solved, where the coefficient matrix $A$ can be expressed as the sum of a matrix $B$, for which efficient solvers are available, and a low-rank matrix $H$. In such cases the Morrison formula [6] can be used to exploit the structure of $B$ and $H$.

This situation occurs when solving some differential problems by finite difference discretization, and the use of Morrison formula is known as Capacitance Matrix Method [1,9].

In this paper the VLSI implementation of Capacitance Matrix Method is investigated. Special attention is given to the solution of Poisson and Helmholtz problems on irregular regions.

Let $A$ be a $q \times q$ nonsingular matrix. Let $A = B + H$, where $B$ is nonsingular and $H$ is of rank $p$, $p < q$, i.e. $H = F G$, where $F$ and $G$ are of size $q \times p$ and $p \times q$, respectively. Then

\begin{equation}
A^{-1} = B^{-1} - B^{-1} F (I + G B^{-1} F)^{-1} G B^{-1}, \tag{1.1}
\end{equation}

where the $p \times p$ matrix $C = I + G B^{-1} F$ is nonsingular and is called the Capacitance Matrix. The solution $x = A^{-1} b$ can be written as

\begin{equation}
x = B^{-1} b - B^{-1} F (I + G B^{-1} F)^{-1} G B^{-1} b. \tag{1.2}
\end{equation}

In section 2 the VLSI implementation of an algorithm to compute (1.2) is studied and the corresponding $AT^2$ bound is given.

In section 3 the above results are applied to the VLSI solution of two differential problems, namely (i) ordinary differential equations with variable coefficients and periodic boundary conditions and (ii) Poisson or Helmholtz partial differential equations on irregular regions.

2. Area-time upper bounds for the Capacitance Matrix Method

In this section a VLSI design is proposed for the implementation of the Capacitance Matrix Method. It is easy to see that the computation of (1.2) can be decomposed into the following steps:

1) Solve the $p$ linear systems $Bz = f$.
2) Compute $C = Gz + 1$.
3) Compute $C^{-1}$.
4) Solve the linear system $Bu = b$.
5) Compute $v = Gu$.
6) Compute $t = C^{-1} v$.
7) Compute $s = F t$.
8) Solve $Br = s$.
9) Compute $x = u - r$.

This algorithm leads to the design of fig. 1, where module I is devoted to solve linear systems with $B$ as coefficient matrix (steps 1, 4, 8); module II performs the matrix-vector product of size $q \times p$ (step 7) and it is
FIG. 1

Module III is capable of storing and shifting out the columns of the matrix \( F \). Module IV performs the matrix-vector product of size \( p \times q \) (steps 2, 5) and, when step 2 is executed, it adds to the results the columns of the identity matrix. Module IV computes the inverse of a \( p \times p \) matrix (step 3), stores the result, and then solves the associated linear system by performing a \( p \times p \) matrix-vector product (step 6). Finally, in module V the partial result of step 4 is stored and the required solution is computed (step 9).

The algorithm can be divided into three phases:

1) A preprocessing phase not depending on the right-hand-side vector, in which the inverse of the capacitance matrix is computed (steps 1-3). This phase is evidenced in fig. 2.

2) The solution of the system \( Bw = b \), and the computation of vector \( v \) (steps 4-5). This phase is evidenced in fig. 3.

3) A final phase in which the solution \( x = w - B^{-1}FC^{-1}v \) is computed, by solving another linear system with coefficient matrix \( B \) (steps 6-9). See fig. 4.

In fig. 5 the VLSI circuit implementing the above described algorithm is presented.

Modules II and III are implemented by using the well-known structure of mesh of trees. An \( r \times s \) matrix-vector product can be performed in \( \lceil \log r \rceil + \lceil \log s \rceil + 1 \) time units by a mesh of trees whose layout has height \( h = O(r \log s) \) and width \( t = O(s \log r) \) [8].

Let \( A_B \) and \( T_B \) be the area and the time used by module B to solve a linear system having \( B \) as coefficient matrix, respectively. Then \( pT_B \) steps suffice to solve the \( p \) systems at step 1 and \( p(\lceil \log q \rceil + \lceil \log p \rceil + 1) \) steps suffice to compute the capacitance matrix.
Module IV can be implemented as a systolic array. Since $p \times p$ matrix inversion and $p \times p$ matrix-vector multiplication can be computed with area $O(p^2)$ and time $O(p)$ [7], then the whole algorithm can be implemented with

$$A = O(A_B + pq \log p \log q + p^2),$$

$$T = O(pT_B + p \log q).$$

Note that the preprocessing phase shown in fig.2 can be performed in time $T_P = O(pT_B + p \log q)$. At the end of this phase, the inverse matrix $C^{-1}$ is available in the circuit and it can be used to solve any system of the type $Ax = b$ in time $T_A = O(T_B + \log q)$.

3. Applications

An immediate application of the previous algorithm is the solution of linear systems derived from the finite difference discretization of the following one dimensional differential problem with periodic boundary conditions, and variable coefficients:

$$\begin{cases}
\alpha(x) \frac{d^2}{dx^2} u(x) = f(x), \\
 u(x_0) = u(x_n).
\end{cases}$$

In this case the discretization leads to the linear system $Ax = b$ where

$$A = \begin{pmatrix}
2\alpha_0 & -\alpha_0 & -\alpha_0 & & \\
-\alpha_1 & 2\alpha_1 & -\alpha_1 & & \\
& -\alpha_2 & 2\alpha_2 & -\alpha_2 & \\
& & \ddots & \ddots & \ddots \\
& & & -\alpha_n & 2\alpha_n & -\alpha_n & \\
& & & & -\alpha_{n-1} & 2\alpha_{n-1} & \alpha_{n-1}
\end{pmatrix}, \quad \alpha_i = \alpha(x_i)$$

$$A = B + H,$$ $B$ being the tridiagonal part of $A$, and $H = FG$ being a rank two $nxn$ matrix, namely

$$F = \begin{pmatrix}
1 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 1
\end{pmatrix}, \quad G = \begin{pmatrix}
0 & 0 & \cdots & -\alpha_0 \\
0 & 0 & \cdots & 0 \\
-\alpha_{n-1} & 0 & \cdots & 0
\end{pmatrix}.$$

It is known that the solution of an $nxn$ tridiagonal linear system can be efficiently obtained by using the odd-even reduction algorithm. In [3] the VLSI implementation of this algorithm was given with $O(n \log n)$ area and $O(\log n)$ time. Therefore, in this case, the VLSI circuit of fig. 5 produces the bounds

$$A = O(n \log n)$$

$$T = O(\log n).$$

Note that in the special case $\alpha(x) = \alpha$, the resulting linear system has a circulant coefficient matrix [5]. Therefore it can be solved in VLSI by using two DFT’s of order $n$ and $o(n)$ arithmetic operations. However this technique yields an $AT^2$ bound of order not less than $O(n^2 \log^2 n)$.

The algorithm shown in section 2 can be efficiently used to obtain the numerical solution of the Helmholtz equation

$$\begin{align}
(3.1) \quad -\frac{\partial^2}{\partial x^2} u(x,y) - \frac{\partial^2}{\partial y^2} u(x,y) + c u(x,y) = f(x,y)
\end{align}$$

on an irregular region $\Omega$ with Dirichlet or Neumann boundary conditions [2].

Let $\mathcal{R}$ be a rectangular region containing $\Omega$. $\mathcal{R}$ is covered by a uniform grid of $m \times n$ points (with $m \approx O(n)$). Let $B$ the coefficient matrix, of order mn, of the linear system derived from the finite difference discretization of the problem (3.1) on $\mathcal{R}$. Let $p$ be the number of "irregular"
mesh points, that is the mesh points in $\Omega$ which have at least one nearest neighbor mesh point outside $\Omega$. It is easy to see that if $\Omega$ is enough regular (e.g. convex) then $p = O(n)$. The matrix $A$, corresponding to the difference Helmholtz problem enlarged to $K$, handling the given boundary conditions on $\partial \Omega$, differs from matrix $B$ only in the $p$ rows corresponding to the irregular mesh points [9].

Let us denote with $G$ the $p \times mn$ matrix being the compact representation of $B - A$, from which the zero rows corresponding to the regular mesh points have been deleted, and let $F$ be the $mn \times p$ matrix obtained selecting in the identity matrix the $p$ columns corresponding to the irregular mesh points. Then $A = B + FG$ and the Capacitance Matrix Method can be applied.

In this case module I results to be the VLSI Poisson solver described in [4], with area $A_0 = O(n^2 \log^2 n)$ and time $T_0 = O(n \log n)$, so that the VLSI circuit of fig. 5 produces the bounds $A = O(n^2 \log^2 n), T=O(n \log n)$.

REFERENCES