Arithmetic Processor for Solving Tridiagonal Systems of Linear Equations

Miloš D. Ercegovac
Computer Science Department
Univ. of California at Los Angeles

Jean-Michel Muller
CNRS-Laboratoire CNRS-ENSL-INRIA-UCBL LIP,
Ecole Normale Superieure de Lyon, France

***Abstract*— We present a method and organization of an arithmetic array processor for solving tridiagonal systems of linear equations. The method uses online arithmetic approach which allows parallel computation of the result digits of the solution vectors. The basic operators are digit-vector by digit multiplication and redundant addition which results in precision-independent cycle time. The method takes about m carry-free cycles to obtain m digits of the solutions. Details of a processor array organization implementing the method and a comparison with a conventional approach are discussed.**

INTRODUCTION

Tridiagonal (TD) systems are frequently used in numerical methods [2], [7]. Examples include finite-difference approximations to partial differential equations and interpolation with splines. Besides using sequential algorithm such as the Thomas' algorithm [11], many parallel TD system solvers have been developed for supercomputers of array and pipeline type, the main approaches being cyclic odd-even reduction and recursive doubling methods [8], [1], [9], [10]. All these solvers were implemented in software. Various scheme were proposed to alleviate corresponding communication problems between processors and memories. This paper presents a direct hardware-oriented approach for solving TD systems intended for application-specific architectures or accelerator coprocessors. Of particular interest are reconfigurable architectures implemented with FPGAs and softcore processors allowing efficient implementation of the primitive arithmetic operations used by the proposed method.

I. THE PROPOSED APPROACH

We solve the linear diagonally-dominant system L

$$L : \mathbf{A} \cdot \mathbf{y} = \mathbf{b} \tag{1}$$

iteratively using MSDF serial arithmetic [3], [6], known as the E-method.

The coefficient matrix $\mathbf{A}$ of order N is

$$A = \begin{bmatrix} 1 & a_{0,1} & 0 & 0 & 0 & 0 \\ a_{1,0} & 1 & a_{1,2} & 0 & 0 & 0 \\ 0 & a_{2,1} & 1 & a_{2,3} & 0 & 0 \\ 0 & 0 & a_{3,2} & 1 & a_{3,4} & 0 \\ 0 & 0 & 0 & a_{4,3} & 1 & a_{4,5} \\ 0 & 0 & 0 & 0 & a_{5,4} & 1 \end{bmatrix}$$

For convergence, the coefficient matrix has to be diagonally dominant as indicated above, i.e.,

$$\sum_{j \neq i} |a_{i,j}| < 1$$

with more specific constraints given later in the paper.

The solution vector is $\mathbf{y} = (y_1, \ldots, y_N)$ and the right-hand side vector is $\mathbf{b} = (b_1, \ldots, b_N)$.

The algorithm for solving the system L is

1. $[Initialize]$
   $\mathbf{w}[0] = \mathbf{b}$; $\mathbf{d}[0] = \mathbf{0}$;
2. $[Recurrence]$
   for $j = 0 \ldots m-1$
   $\mathbf{v}[j] = r(\mathbf{w}[j] - \mathbf{A}\mathbf{d}[j])$;
   $\mathbf{d}[j+1] \leftarrow SEL(\widehat{\mathbf{v}}[j])$;
   $\mathbf{w}[j+1] \leftarrow \mathbf{v}[j]$;
   $\mathbf{y}[j+1] \leftarrow CONVERT(\mathbf{y}[j], SEL(\widehat{\mathbf{v}}[j]\ ))$
   end for
3. $[Result]$
   $\mathbf{y}\ [m]$

where:

- The output digit selection function uses rounding of $\widehat{vk[j]}$, an estimate of $vk[j]$ as argument, truncated to one fractional bit:

  $$d_{k(j+1)} = SEL(\widehat{vk[j]}) = \left\lfloor \widehat{vk[j]} + \frac{1}{2} \right\rfloor$$

  For radix 2 case, the selection function reduces to

  $$\begin{cases} 1 & \textbf{if } \widehat{vk[j]} \geq 0.5 \\ 0 & \textbf{if } -0.5 \leq \widehat{vk[j]} \leq 0 \\ -1 & \textbf{if } \widehat{vk[j]} \leq -1 \end{cases}$$

- The residual vector at step j is

  $$\mathbf{w}[j] = (w1[j], \ldots, wN[j])$$

- The result digit-vector at step j is

$$\mathbf{d}[j] = (d1[j], \ldots, dN[j])$$

where digit $dk_j \in \{-1, 0, 1\}$ is the j-th digit of the k-th solution component

$$y_k = \sum_{j=1}^{m} dk_j 2^{-j}$$

- $CONVERT$ performs on-the-fly conversion [6] of the redundant result digits produced serially into a conventional digit-parallel form.

We now discuss the convergence conditions of the algorithm. For simplicity, we focus on radix-2 iterations. Adaptation to higher radices is straightforward. The iterations converge to the desired result if vector $\mathbf{w}[j]$ is bounded. Define constants ξ, α and Δ (the overlap between the selection intervals $0 \leq \Delta < 1$) such that

$$\sum_{j \neq i} |a_{i,j}| \leq \alpha \tag{2}$$

for each row of the coefficient matrix $\mathbf{A}$ and

$$\begin{cases} |b_k| & \leq \xi \\ |wk[j] - \widehat{wk[j]}| & \leq \frac{\Delta}{2} \end{cases}$$

Since $|d_{k(j-1)} - \widehat{wk[j-1]}| \leq 1/2$ due to the rounding rule of the selection function, from the residual recurrence we find

$$|wk[j]| \leq 2\left(\frac{1}{2} + \frac{\Delta}{2} + \alpha\right) = 1 + \Delta + 2\alpha. \tag{3}$$

To achieve this bound, we must assure that a suitable choice of d_{kj} in $\{-1, 0, 1\}$ is possible. This implies that $|wk[j]|$ should not be larger than $3/2$, giving immediately the following condition

$$\Delta + 2\alpha \leq \frac{1}{2} \tag{4}$$

For $\Delta = 0$ (no overlap, non-redundant residual), the bound on the sum of off-diagonal elements (absolute value) in a row is $\alpha = 1/4$. For $\Delta = 1/4$, $\alpha = 1/8$. Since $|w_k[0]|$ must not be larger than $3/2$, we get for the bound on the initial values

$$\xi \leq \frac{3}{2} \tag{5}$$

The E-method has the following features:

- Each solution component y_k is evaluated on a separate MSDF multiply-add module.
- The module uses a digit-vector $\times$ digit multiplication and a redundant addition, i.e., carry-save or signed-digit form.
- The method eliminates the use of explicit division.
- For higher radix r, which has stricter convergence conditions, suitable prescaling is used.

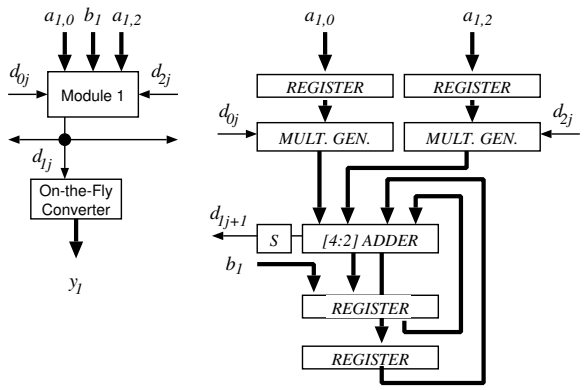

Fig. 1. Basic module block diagram and organization.

A. Basic Module

The basic module implements the residual recurrence

$$w1[j+1] \leftarrow r(w1[j] - a_{1,0}d_{0j} - a_{1,2}d_{2j} - d_{1j+1}) = v1[j] \tag{6}$$

(for $i = 1$) and the corresponding digit selection function

$$d_{1(j+1)} = SEL(\widehat{v1[j]}) \tag{7}$$

The module organization is shown in Figure 1.

It consists of a [4:2] adder, two multiple generators which are implemented as digit vector by digit multipliers, and four registers. To avoid carry-propagation in multiple generators, the output can be obtained in a redundant form (e.g., sum-carry vectors) and the residual adder becomes a [6:2] adder. The output digit is selected as mentioned above. There are also four registers storing two coefficients, and a sum-carry representation of the residual. In the case of radix 2, the multiple generators are reduced to multiplexers.

B. General Scheme for a TD System Solver

Figure 2 depicts a general organization of a TD solver: for an N-the order system, N basic modules are used. These modules have two digit-serial inputs, connected to the left and right adjacent modules (except the first and the last module), one digit-serial output, and two parallel ports per module to load the coefficients. The output from each module is in a redundant form which, if needed, can be converted using on-the-fly converters into a conventional 2's complement parallel form. There is no extra delay of this conversion, i.e., it is performed in parallel with the main digit recurrence.

The approach allows easy extension of the order of the system being solved or/and of the precision. If modules are augmented by a $2k$-register FIFO, the N module scheme can implement a solver for a TD system of order kN, or, alternately, handle precision k times the precision of the basic module. In either case, the recurrence cycle cycle is k times longer than the basic module cycle.

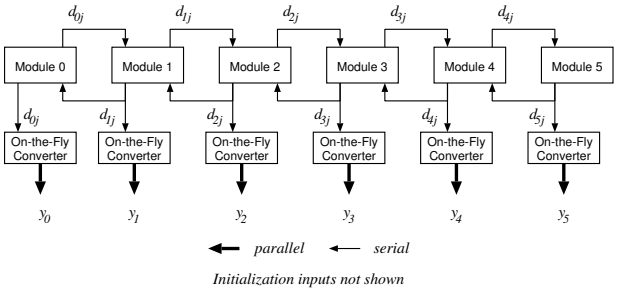

Fig. 2. General Scheme for a TD Solver.

II. Example 1: Online Implicit Method for Solving PDEs

As an example of the application, we present the use of the proposed approach in implementing the implicit method for solving partial differential equations, originally developed in [12]. The parabolic equation with appropriate initial and boundary conditions

$$\bigtriangledown^2 \Phi = k \frac{\partial \Phi}{\partial t} \tag{8}$$

can be solved by implicit methods in which derivatives with respect to time are approximated by backward differences. These methods are always computationally stable. However, such a method requires solving a sparse system of linear equations. We show how to use the E-method in solving this system. The finite difference equation for system (8) at the point $(i,\ t)$ is

$$\frac{1}{\Delta x^2}(\Phi_{i-1,t} - 2\Phi_{i,t} + \phi_{i+1,t}) = \frac{k}{\Delta t}(\Phi_{i,t} - \Phi_{i,t-\Delta t} \tag{9}$$

Arranging (9) leads to the following finite difference system for the E-method

$$-p\Phi_{i-1,t} + \Phi_{i,t} - p\Phi_{i+1,t} = q\Phi_{i,t-\Delta t} \tag{10}$$

where $p = (2 + \frac{k\Delta x^2}{\Delta t})^{-1}$, $q = p\frac{k\Delta x^2}{\Delta t}$ and $0 \le t \le N\Delta t$.

This system corresponds to the following tridiagonal system of linear equatuions suitable for the application of the E-method:

$$\begin{bmatrix} 1 & -p & 0 & 0 & 0 & 0 \\ -p & 1 & -p & 0 & 0 & 0 \\ 0 & -p & 1 & -p & 0 & 0 \\ 0 & 0 & -p & 1 & -p & 0 \\ 0 & 0 & 0 & -p & 1 & -p \\ 0 & 0 & 0 & 0 & -p & 1 \end{bmatrix} \begin{bmatrix} \Phi_{1,t} \\ \Phi_{2,t} \\ \Phi_{3,t} \\ \Phi_{4,t} \\ \Phi_{5,t} \\ \Phi_{6,t} \end{bmatrix}$$

$$= q \begin{bmatrix} \Phi_{1,t-\Delta t} \\ \Phi_{2,t-\Delta t} \\ \Phi_{3,t-\Delta t} \\ \Phi_{4,t-\Delta t} \\ \Phi_{5,t-\Delta t} \\ \Phi_{6,t-\Delta t} \end{bmatrix} + p \begin{bmatrix} \Phi_{b1} \\ 0 \\ 0 \\ 0 \\ 0 \\ \Phi_{bn} \end{bmatrix}$$

where Φ_{b1}, Φ_{bn} are the boundary conditions. For the convergence of the E-method it was established that $p \le 0.2$,which can be achieved by $\Delta t \le k\Delta x^2/3$ [12].

The expressions on the right-hand side are computed using two left-to-right carry-free (LRCF) multipliers [4] and online adder [6], combined into OMAU module with a total online delay $\delta_{RHS} = 3+2$ for $r = 2$. The TD solver uses a modified E-method in which the right-hand side elements are used in the MSDF manner [12], [5]. Its online delay is $\delta_E = 2$. The total online delay for one time sweep is $\Delta = 7$ (plus 1 cycle to output the result digit). For a fully unfolded implementation which has the lowest latency, we need $\lceil m/8 \rceil \times n$ OMAUs and E-method basic modules. Figure 3 illustrates the overall scheme.

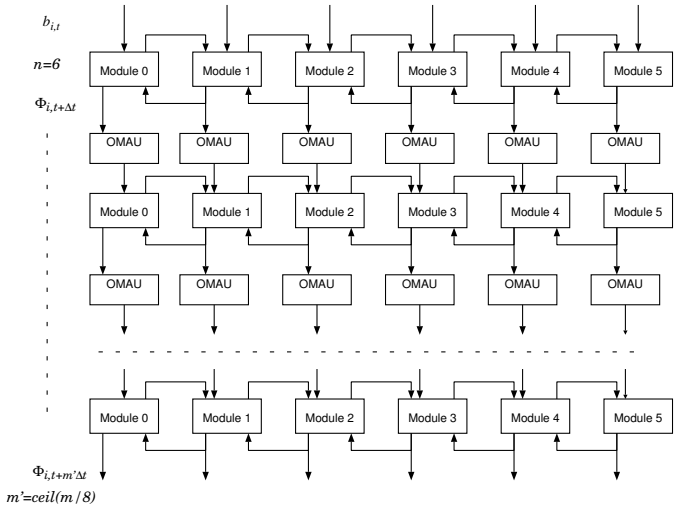

Fig. 3. Unfolded Implementation of Parabolic Equation Solver

The cycle time t_{cycle} of the recurrence loop of the E-method, measured in full-adder delays (t_{FA}), is estimated as

$$t_{cycle} = t_{SEL} + t_{MG} + t_{[4:2]} + t_{reg} = 4t_{FA}$$

is larger than the OMAU cycle. The component delays are:

- $t_{SEL} = 1.5t_{FA}$ is the delay of the selection function,
- $t_{MG} = 0.5t_{FA}$ is the delay of the multiple generator network,
- $t_{[4:2]} = 1.5t_{FA}$ is the delay of the redundant adder,
- $t_{reg} = 0.5t_{FA}$.

The overall delay is estimated as

$$T(N,n,m) = [(\delta+1)(N-1)+m+1]t_{cycle} \approx (8N+m)t_{cycle} \tag{11}$$

The time of a sequential implementation of the Thomas' algorithm for solving tridiagonal systems [11], assuming the same cycle time for all operations, is

$$T_S(N,n,m) \approx (7n+4)mNt_{cycle} \tag{12}$$

Consequently, the proposed scheme has a speedup $O(mn)$ with respect to the Thomas' sequential algorithm. Clearly, the cost of the proposed scheme is much higher than that of a

sequential implementation. Details of the cost comparison will not be discussed here.

III. Example 2: Solving Tridiagonal Systems with Special Coefficient Matrices

The E-method is particularly efficient in solving tridiagonal systems that have coefficient matrix consisting of simple fractions. For example, consider the following system:

$$\begin{bmatrix} 1 & 1/4 & 0 & 0 & 0 & 0 \\ 1/8 & 1 & 1/8 & 0 & 0 & 0 \\ 0 & 1/8 & 1 & 1/8 & 0 & 0 \\ 0 & 0 & 1/8 & 1 & 1/8 & 0 \\ 0 & 0 & 0 & 1/8 & 1 & 1/8 \\ 0 & 0 & 0 & 0 & 1/4 & 1 \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \\ k_4 \\ k_5 \end{bmatrix} = \begin{bmatrix} b_0, b_1, b_2, b_3, b_4, b_5 \end{bmatrix}^T$$

The residual recurrence (shown for $i = 1$)

$$\begin{aligned} w1[j+1] &\leftarrow r(w1[j] - d_{0j}2^{-3} - d_{2j}2^{-3} - d_{1(j+1)}) \\ &= v1[j] \end{aligned} \quad (13)$$

is very simple and leads to a greatly simplified implementation of the basic module: a residual in a nonredundant (2's complement) form is sufficient; the adder for radix 2^k is $k+1+3$ bits wide. For $r = 2$, the adder can be replace by an 8-input, 2-output combinational network.

The simplified module implementation for radix 2 case is shown in Figure 4.

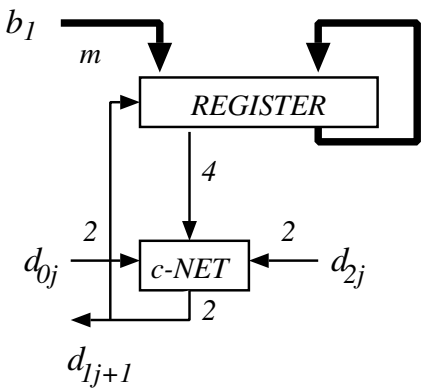

Fig. 4. Module for Special TD System

We estimate the cycle time as

$$t_{cycle} = t_{c-net} + t_{reg} \approx (0.5 + 0.5)t_{FA} = 1t_{FA} \quad (14)$$

and the total time to solve the TD system as

$$T \approx mt_{FA} \quad (15)$$

The cost of a module is roughly

$$\begin{aligned} C_M &= 2 \times C_{c-net} + (m+1)C_{FF} \\ &\approx (2 + (m+1)/2)C_{FA} \approx (3+m)C_{FA} \end{aligned} \quad (16)$$

and the total cost of the solver for $N = 6$ shown in Figure 5 is

$$C \approx (6m + 18)C_{FA} \quad (17)$$

which is much less than a cost of a single $m \times m$ multiplier. The cost of on-the-fly converters is about $6 \times mC_{FA}$.

The TD solver for $N = 6$ is shown in Figure 5.

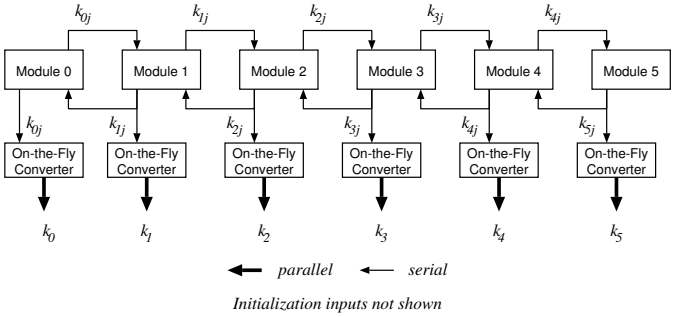

Fig. 5. A Solver for Special TD System

IV. Summary

We have investigated online algorithms for a highly parallel TD solver which consists of a linear array of basic modules. Each basic module implements an online multiply-add operator with a cycle time independent of precision due to the redundancy in representation of the residuals. The latency is m cycles for m digits of precision, i.e., equivalent to the latency of a serial-parallel multiplier. The basic module has a simple implementation and it communicates using digit-serial method. This is advantageous in physical realization. The proposed approach allows also simple mapping of larger TD systems or/and larger precision on a fixed array of given precision. The approach is suitable for variable precision as well as for compound algorithms. The work in progress include mapping of a TD solver to FPGA platforms with soft cores which allow custom instruction sets. Such a set of special instructions implementing primitive operations of the E-method would simplify the use of the proposed arithmetic processing approach.

References

[1] B.L. Buzbee, G.H. Golub, and C.W. Nielson, "On Direct Methods for Solving Poisson's Equations,", *SIAM J. Numer. Anal.*, 7(4):627-656, December 1970.

[2] G. Dahlquist and A. Bjorck, *Numerical Methods*, Englewood Cliffs, NJ, Prentice-Hall, Inc., 1974

[3] M.D. Ercegovac, "A General Hardware-Oriented Method for Evaluation of Functions and Computations in a Digital Computer", *IEEE Transactions on Computers*, C-26(7):667-680, July 1977.

[4] M.D. Ercegovac and T. Lang. Fast multiplication without carry-propagate addition. *IEEE Trans. Comput.*, C-39(11):1385–1390, November 1990.

[5] M.D. Ercegovac, J.M. Muller, and A. Tisserand, "FPGA implementation of polynomial evaluation algorithms." *Proc. SPIE on Field Programmable Gate Arrays (FPGAs) for Fast Board Development and Reconfigurable Computing*, volume 2607, pages 177–188, 1995.

[6] M.D. Ercegovac, T. Lang, *Digital Arithmetic*, San Francisco, Morgan Kaufmann, 2004.
[7] G.H. Golub and C. Van Loan, *Matrix Computations*, 2nd Edition, Baltimore and London, Johns Hopkins University Press, 1989.
[8] R.W. Hockney, "A Fast Direct Solution of Poisson's Equation Using Fourier Analysis," *J. ACM*, 12:95-113, 1965.
[9] H.S. Stone, "An Efficient Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations," *J. ACM*, 20:27-38, 1973.
[10] H.S. Stone, "Parallel Tridiagonal Equation Solvers," *ACM Trans. Math. Soft.*, 1:289-307, 1975.
[11] L.H. Thomas, "Elliptic Problems in Linear Difference Equations over a Network," Watson Sci. Comput. Lab. Rept., Columbia University, New York, 1949.
[12] O. Watanuki, "Fast Parallel Solution of Partial Differential Equations: Application of On-Line Methods to PDE Solvers," UCLA Computer Science Department, Internal Report, 1979.