An Efficient FPGA-Based Direct Linear Solver

Zhenhua Jiang, *Senior Member, IEEE*
Energy Technologies and Materials Division
University of Dayton Research Institute
Dayton, OH, USA
Email: zjiang4@udayton.edu

Sayed Ata Raziei, *Student Member, IEEE*
Electrical and Computer Engineering
University of Dayton
Dayton OH, USA
Email: razieis1@udayton.edu

***Abstract*— This paper presents a novel method to finding the solution to a system of linear equations efficiently by using a reconfigurable hardware based real-time computational solver. The presented linear solver is to directly solve the system of linear equations through repetitively applying Gauss-Jordan elimination to each column of an augmented matrix in parallel on reconfigurable hardware, which can greatly accelerate the solution procedure. Backward substitution is not needed, so the computing latency can be further reduced. The main components of the hardware solver include parallel data processing modules, reusable memory blocks and flexible control logic units. By considering pivoting, this solver can avoid the potential problem of increasingly-large numbers after row operations. The salient feature is that the latency of this solver is really low through parallel processing, deep pipelining and flexible use of memory blocks. For instance, the total latency of this linear solver is controlled below 1000 clock cycles for a dense system of dimension 32. On a Xilinx Vertex 6 FPGA of 200MHz, which has a clock cycle of 5ns, the minimum latency can be as low as 5 microseconds. Applications of this hardware accelerated linear solver may include, but are not limited to, real-time least square estimation for sensor data, digital signal / video processing and real-time circuit simulation. It can also find wide applications in mathematical computing such as finding the inverse of a matrix, computing determinants or ranks of matrices, etc.**

Keywords—FPGA; systems of linear equations; direct linear solver; Gauss-Jordan elimination; real-time computing

I. INTRODUCTION

Nowadays, the need of finding the solution to a large set of simultaneous linear equations can be generally found in a vast variety of scientific and engineering problems. A linear solver, which aims to find the solution on a digital computing system, can achieve such a goal easily and find wide applications in the areas such as least square regression for sensor data [1], digital signal/video processing [2], nonlinear model predictive control [3], real-time circuit simulation [4], etc. It can also be used in mathematical computing such as finding the inverse of a matrix, computing determinants or ranks of matrices, etc.

Matrix operations are a common task for digital computers. A system of linear equations can be easily re-arranged into a matrix form, where each equation becomes a row in the matrix. While all variables can be assembled and placed into a column vector, $\boldsymbol{x}$, the coefficients associated with the variables in all equations will constitute a matrix, $\boldsymbol{A}$. If an additional column vector, $\boldsymbol{b}$, is added to the right hand side, the system of linear equations can then be represented in the matrix format by

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3N} \\ \dots & \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & a_{N3} & \dots & a_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_N \end{bmatrix}$$

In short, this can be described as $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A}$ is an NxN matrix, $\boldsymbol{b}$ is an Nx1 vector, and $\boldsymbol{x}$ is an Nx1 vector.

In the literature, there exist two major types of methods to solving the linear equation system: direct methods and iterative methods. Direct methods include LU (lower-upper triangular) factorization, QR factorization, Cholesky factorization, etc., which can be typically used for dense linear systems [5]-[7]. Iterative methods may include Jacobi, Gauss-Seidel and relaxation iterations, which are suitable for sparse linear systems [8]-[10]. This work considers a direct linear solver and its real-time hardware implementation which aims to accelerate the solution process by use of parallelism and pipelining.

A popular direct method is to use the Gaussian elimination algorithm [11]. The Gaussian elimination procedure updates the matrix continuously by applying a sequence of basic row operations to the lower portion of the matrix until the lower left-hand corner of the matrix becomes filled with zeros. Three types of basic row operations include swapping two rows, multiplying a row by a coefficient, and subtracting a multiple of one row from another row. Following a series of operations, a matrix can be transformed into an upper triangular matrix. A backward substitution process can then be applied to find the solution in sequence based on the upper triangular matrix.

A further-reduced method, called Gauss-Jordan Elimination, is to eliminate one column in all rows except the pivot value or diagonal entry within each iteration loop, which can be set to 1 by normalizing each row [12]. The resultant updated matrix after all elimination iterations will be a unit diagonal matrix, so the backward substitution is not needed. The solution vector will be the last column of the updated augmented matrix, i.e., the solution value in each row is the last value of that particular row vector. This method is not preferred in software solution because it involves more arithmetic operations than traditional Gaussian elimination method. However, on parallel processing hardware, the Gauss-Jordan method is more efficient because the hardware allows for parallel processing and elimination of all rows simultaneously without increasing the processing time.

Field programmable gate arrays (FPGAs) are one type of semiconductor IC devices that consist of a large number of

This work was sponsored by the "Ohio Research Scholar" funding of the "Ohio Third Frontier" Program.

reconfigurable logic units, many programmable input/output blocks and interconnects [13]-[14]. Since those programmable logic arrays on FPGAs are massively-parallel units, they naturally allow for parallel processing of a large amount of data. Clock signals generated from high-frequency oscillators enable data processing and operation with a clock cycle of as low as 5 ns. Nowadays, user-friendly, high-level programming tools, such as Xilinx System Generator [15], are available to be used to program these reconfigurable logic devices, which can help reduce the product development time. Another benefit is that FPGAs perform deterministic computing, which makes timing control easy, especially good for real-time implementation.

This paper presents an efficient method for the hardware implementation of the direct linear solver based on FPGAs. The presented linear solver is to directly solve the system of linear equations through repetitively applying Gauss-Jordan elimination to each column of an augmented matrix. The hardware accelerator can exploit the inherent parallelism in the algorithm of finding the solution, offering an efficient implementation of the linear solver. The specific method and algorithm flow chart will be discussed in Section II. Section III will elaborate on the functionality and detailed implementation circuit of each portion of the proposed linear solver. Section IV will present two application examples of the linear solver, focusing on the computational latency and resource utilization. Section VI will conclude the paper and outline the future work.

II. Efficient Method to Finding Direct Solution to Systems of Linear Equations

A. Gauss-Jordan Elimination Procedure

Fig. 1 illustrates the procedure about how the Gauss-Jordan Elimination method, which is the foundation of the proposed FPGA-based linear solver, can be used to solve the linear system of equations. To facilitate the solution process and parallel processing, a column-augmented matrix is formed, i.e., [***A*** | ***b***], by concatenating ***A*** and ***b***, as illustrated in Fig. 1-a. The row operations are then based on this augmented matrix [***A*** | ***b***], where the number of columns is the number of rows plus one. As demonstrated in Fig. 1-b, the main tasks of the repetitive elimination in the Gauss-Jordan Elimination procedure include the following three steps:

Pivot search: This step is to search the largest element, at the current iteration, in a lower column that is below the pivot element found and normalized at a previous iteration. As an example, at the 4th iteration, the lower column may include the elements below the entry associated with Row 3 (i.e., 1) in the 4th column. Upon completion of the search process which can be done in parallel and with pipelining, the maximum value is selected as the pivot element. By considering pivoting, the potential problem of increasingly-large numbers following a series of row operations could be avoided.

Row swap: This step is to exchange the found row with the previous *k*th row. The row with the maximum value becomes the new base row and the maximum value is placed in the pivot entry. This operation can be achieved by updating the index of the row vectors in the matrix, where the index is stored in a separate vector. The practical movement of data in the memory is not needed, thus the latency is greatly reduced.

Row update: This step is to update the right-hand side of the augmented matrix in all the rows with the goal that the pivot column will be replaced with one in the pivot entry and zeros in all other rows. This step involves a major part of arithmetic operations and can be done in parallel with efficient memory and arithmetic units.

Within each iteration loop, an unknown variable in the linear system of equations (i.e., corresponding to a column in the matrix) can be eliminated. By the end of *N* iterations, the matrix ***A*** becomes a unit diagonal matrix; therefore, the ***b*** vector becomes ***x***, as shown in Fig. 2-c.

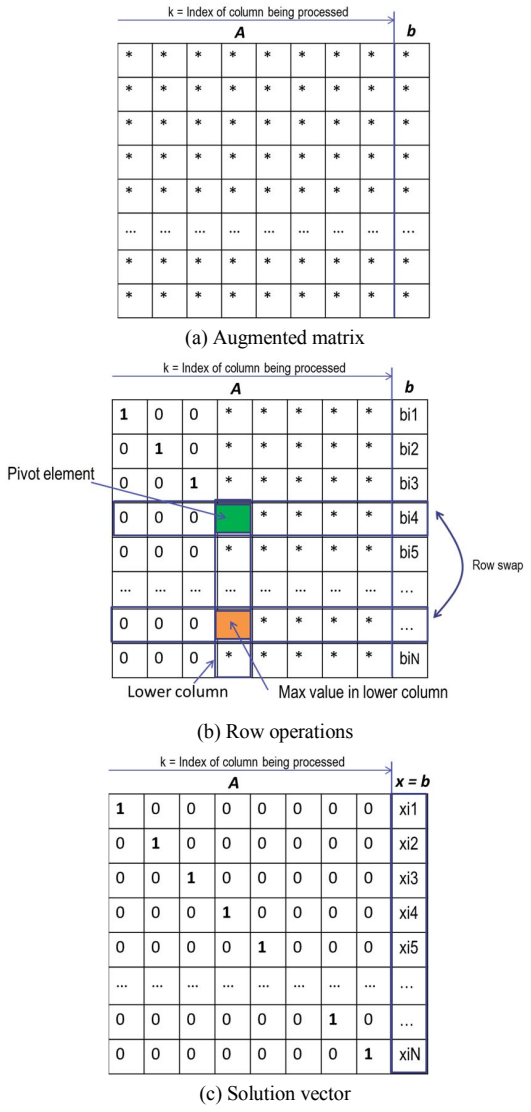

Fig. 1. Illustration of the Gauss-Jordan Elimination procedure.

B. Algorithm Flow Chart

Based on the principles of the procedure explained above, Fig. 2 shows the flow chart of the Gauss-Jordan elimination algorithm in solving a linear system of equations. The main steps in the flow chart are explained as follows. The first step is to receive the input data in the format of an augmented matrix