Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 1 (2012) 31 – 35

Procedia Technology

INSODE 2011

Using gauss - Jordan elimination method with CUDA for linear circuit equation systems

Nesrin Aydin Atasoy[a] *, Baha Sen[b], Burhan Selcuk[c]

[a,b,c] *Karabuk University, Engineering Faculty, Computer Engineering Department,Baliklarkayasi Mevkii,78050,Karabuk,TURKEY*

Abstract

Many scientific and engineering problems can use a system of linear equations. In this study, solution of Linear Circuit Equation System (LCES) for an *nxn* matrix using Compute Unified Device Architecture (CUDA) is described. Solution of LCES is realized on Graphics Processing Unit (GPU) instead of Central Processing Unit (CPU). CUDA is a parallel computing architecture developed by NVIDIA. Linear Circuits include resistance, impedance, capacitance, dependent - independent current sources and DC, AC voltage source. In this study, solutions of circuits that include resistance, independent current sources and DC voltage source have analyzes. Circuit analysis frequently involves solution of linear simultaneous equations that are solved Gauss-Jordan Elimination Method in this study. Gauss-Jordan Elimination is a variant of Gaussian Elimination that a method of solving a linear system equations (*Ax=B*). Gauss-Jordan Elimination is an algorithm for getting matrices in reduced row echelon form using elementary row operations. Gaussian Elimination has two parts. The first part (Forward Elimination) reduces a given system to triangular form. The second step uses back substitution to find the solution of the triangular echelon form system Because of elements of unknowns column matrix are dependent on each other, second step algorithm is not appropriate for parallel programming. Two parts of Gauss–Jordan Elimination are not like Gaussian Elimination's part so it is preferred. GPU implementation is more faster than solution of linear equation systems on CPU.

Keywords: CUDA, GPU Computing, Gauss-Jordan Elimination Method, Linear Circuit Equation Systems

1. Introduction

Many scientific and engineering problems can take the form of a system of linear equations. Because linear systems derived from realistic problems are often quite complex [1].

Linear Equation System (LES) in Equation 1 including m equation and unknown in n number is given below;

$$\begin{aligned} a_{11}x_1 &+ a_{12}x_2 \ldots + a_{1n}x_n &&= b_1 \\ a_{21}x_1 &+ a_{22}x_2 \ldots + a_{2n}x_n &&= b_2 \\ \vdots &\quad\ \vdots &&\ \ \vdots \\ a_{m1}x_1 &+ a_{m2}x_2 \ldots + a_{mn}x_n &&= b_m \end{aligned} \qquad (1)$$

* Nesrin Aydin Atasoy. Tel.: +0-370-433-2021-186; fax: +0-370-433-3290.
E-mail address: nesrinaydin06@yahoo.com

2212-0173 © 2012 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.protcy.2012.02.008

$x_1, x_2, \ldots, x_n$ denote unknowns, a's and b's denote constants. LES is written in matrix form shown in Figure 1 and 2 in order to be solved with appropriate and rapid method.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \qquad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Fig.1. (a Coefficients matrix.; (b) Unknowns column matrix; (c) Constants column matrix.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$$A \qquad X = B$$

Fig.2. Matrix form of LES.

It is possible to classify methods used in the numerical analysis of LESs into two groups respectively as "Elimination Methods (Gauss Elimination, Gauss-Jordan Elimination, etc.)" and "Iterative Methods (Jacobi Iterative, Seidel Iterative, etc.)". Gaussian Elimination is a well-known technique for solving both problems; it consists in transforming a matrix to row echelon form, from which the inverse is derived automatically. Gauss-Jordan (GJ) elimination is an extension of this method, in which the matrix is further reduced to its reduced row echelon form [2].

Circuit analysis frequently involves the solution of linear simultaneous equations [3]. Linear Circuits include resistance, impedance, capacitance, dependent - independent current sources and DC, AC voltage source. Since resistance and DC voltage resource were used within the study, Gauss-Jordan (GJ) elimination method is used within the settings of the study. Unknown current values are found out from the solution of circuit equation system.

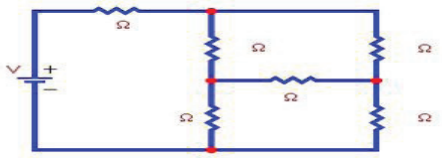

Fig.3. Electronic circuit consisting of three loop parallel and serial resistances.

In this study we try to describe solution of LCES for an *nxn* matrix using CUDA with enabling GPU. Solution of LCES is realized on GPU instead of CPU and the aim is providing acceleration. CUDA is a parallel computing architecture developed by NVIDIA. Recently, GPU has led the advances in computation for science and engineering applications due to highly parallel programming performance, such as massive multithreading and high memory bandwidth, etc. [4].

2. Solution of Linear Circuit Equation Systems Based on Gauss - Jordan Elimination Method Using CUDA

2.1. Gauss - Jordan Elimination Method and CUDA

Gaussian Elimination is a traditional method for solving LESs and finding the inverse of a matrix, in which row operations are used to change the coefficient matrix to the upper triangular echelon form; the solution of the LES is

then found out by back substitution. A more elaborate solution is Gauss-Jordan elimination, in which row operations are used to transform the coefficient matrix to reduced row echelon form (i.e., only elements in the main diagonal have a value of 1, and all other elements 0), therefore, the answer becomes more apparent [2].

Augmented matrix of A matrix in Figure1 (a) is denoted as Equation 2.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} & b_m \end{bmatrix} = [A \vdots B] \tag{2}$$

Augmented matrix is transformed into a matrix whose elements in the main diagonal have a value of 1 via elementary row transformation. The process of obtaining solution of LES by transforming given augmented matrix into an equivalent $[A^* \vdots B^*]$ matrix in Equation 3 mentioned above via elementary row transformations is called as Gauss-Jordan Elimination Method.

$$\begin{bmatrix} 1 & 0 & \dots & 0 & b_1^* \\ 0 & 1 & \dots & 0 & b_2^* \\ 0 & 0 & 1\dots & 0 & b_3^* \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \dots & 1 & b_n^* \end{bmatrix} \tag{3}$$

However, parallelism cannot be applied in back substitution to find unknowns after forming upper triangle as it is seen in Equation 4 used in GEM, because, the values of variables within this calculation are dependent on each other. On the other hand, values of variables within GJEM are not dependent on each other since triangles are formed above and below the diagonal line. In other words, there is no need for back substitution. Therefore, GJEM is preferred within the scope of this study.

$$\begin{bmatrix} 1 & a_{12}^* & \dots & a_{1n}^* & b_1^* \\ 0 & 1 & \dots & a_{2n}^* & b_2^* \\ \vdots & & 1 & \vdots & \vdots \\ 0 & 0 & \dots & 1 & b_n^* \end{bmatrix} \tag{4}$$

GPU programming is preferred instead of CPU in terms of execution time in order to find out solutions of LCES with GJEM. Moreover CUDA is preferred to run the application developed on GPU.

CUDA comes with a software environment that allows developers to use C as a high-level programming language C for CUDA extends C by allowing the programmer to define C functions, called kernels, that, when called, are executed N times in parallel by N different CUDA threads, as opposed to only once like regular C functions [5].

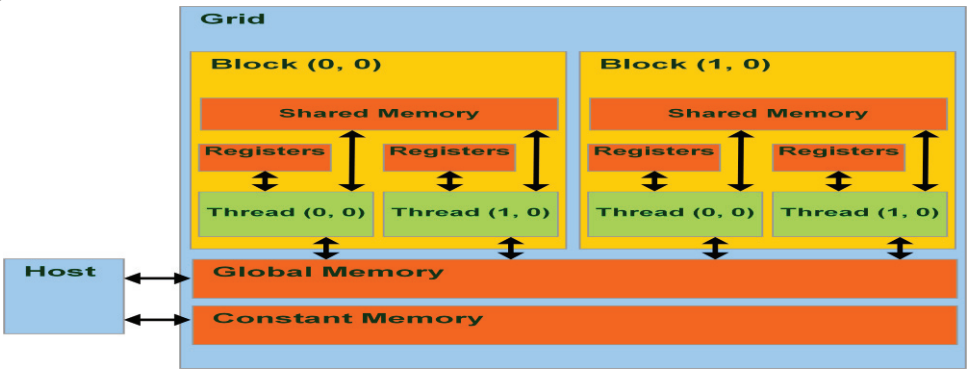

Fig.4. CUDA thread and memory allocation [6].

2.2. Implemented Solution of Linear Circuit Equation Systems on CPU and GPU

Application is developed on a PC with dual core including GeForce GT 240M graphics card utilizing C programming language in Visual Studio 2008. GJEM algorithm can be depicted in matrix form as [7]:

Initial stage

$$\begin{matrix} pivot \\ m_{21}=-1 \\ m_{31}=-1 \end{matrix}\left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & -1 \\ 1 & 3 & 9 & 1 \end{array}\right]$$

Iteration 1

$$\begin{matrix} \\ pivot \\ m_{32}=-2 \end{matrix}\left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 \\ 0 & 1 & 3 & -2 \\ 0 & 2 & 8 & 0 \end{array}\right]$$

Iteration 2

$$\left[\begin{array}{cccc|c} 1 & 0 & -2 & 3 \\ 0 & 1 & 3 & -2 \\ 0 & 0 & 2 & 4 \end{array}\right]$$

Iteration 3

$$\left[\begin{array}{cccc|c} 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & -8 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

As it can be easily seen in the figure above, iterations are needed to form triangles above and below the diagonal line. Furthermore, back substitution cannot be applied within this calculation. It can be said that GJEM is suitable for parallelism.

Startup screen in Figure 5 is displayed when the application is started.

```
MENU
1-Creating augmented matrix file of LCES
2-Equation Solving
3-Exit
Choice: _
```

Fig.5. Startup screen.

Augmented matrix is formed via the input provided (in .txt file) by the user with the selection of menu item 1.

```
MENU
1-Creating augmented matrix file of LCES
2-Equation Solving
3-Exit
Choice: 1
Please,input matrix size: 3
Please,input file name: gpu.txt
matrix[0][0]= 1
matrix[0][1]= 2
matrix[0][2]= 1
matrix[0][3]= 8
matrix[1][0]= 2
matrix[1][1]= -1
matrix[1][2]= 1
matrix[1][3]= 3
matrix[2][0]= 3
matrix[2][1]= 3
matrix[2][2]= -2
matrix[2][3]= 3

File is created succesfully..._
```

Fig.6. Creating matrix form.

Solution stages and results of equation system formed above are displayed with the selection of menu item 2. Whole calculations last 0.0012 msec and the results are as follows respectively: x_1 =1, x_2=2 and x_3 =3.

```
MENU
1-Creating augmented matrix file of LCES
2-Equation Solving
3-Exit
Choice: 2
Please,input matrix size: 3
Please,input file name: gpu.txt

These elements are read from file
1.000000 2.000000 1.000000 8.000000
2.000000 -1.000000 1.000000 3.000000
3.000000 3.000000 -2.000000 3.000000
Sorted matrix=
1.000000 2.000000 1.000000 8.000000
2.000000 -1.000000 1.000000 3.000000
3.000000 3.000000 -2.000000 3.000000 Time to generate:  3.9 ms


Row echelon form

1.000000 0.000000 0.000000 1.000000
0.000000 1.000000 0.000000 2.000000
-0.000000 -0.000000 1.000000 3.000000
```

Fig.7.GPU result screen.

The last stage (menu item 3) is used for exiting from the menu. On the other hand, execution time of CPU code is 5.101000 sec in Figure 8.

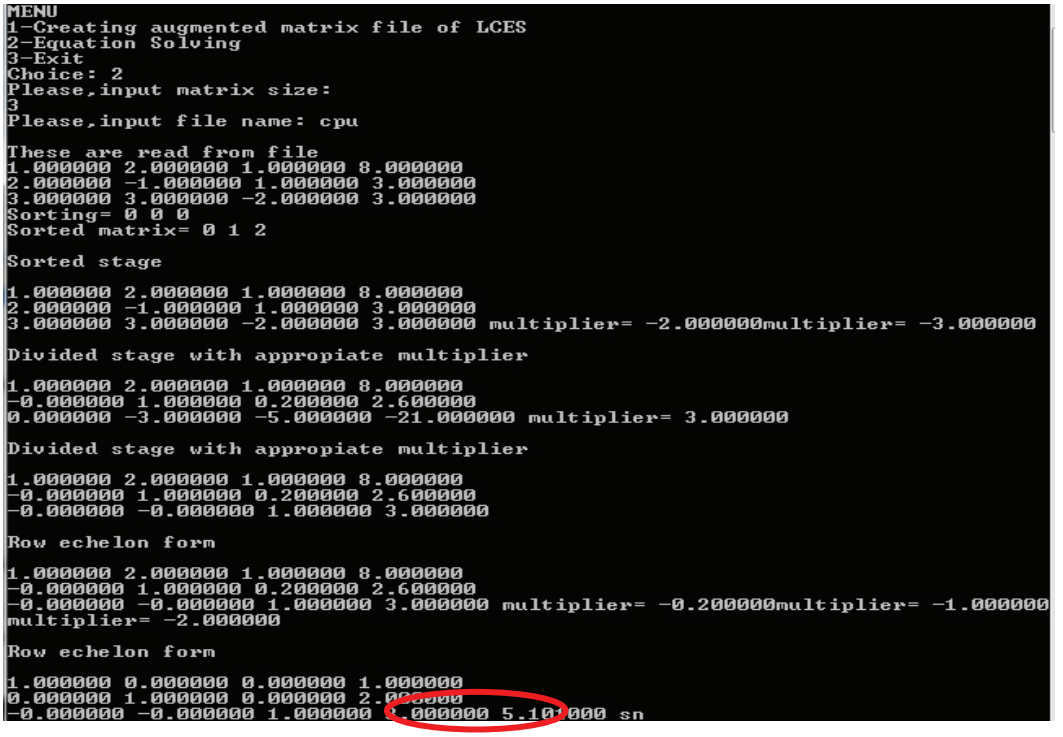

Fig.8.CPU execution time screen.

3. Results and Discussion

Parallel programming techniques are gaining importance from day to day. In this study, the performance of Gauss-Jordan Elimination Method is examined in terms of execution time by using parallel programming techniques on graphic card. Application developed on GPU utilizing CUDA is faster than the application developed on CPU in the solution of *n* unknown linear equation system such as 3x3, 4x4, etc.

Furthermore, both the developments of applications regarding OpenMP parallel programming techniques and studies of applications developed by utilizing CUDA have been still continuing.

References

1. M.J. Quinn, Parallel programming in C with MPI and OpenMP, McGraw-Hill Higher Education, Singapore, International Edition 2003, pp.90.
2. An Improved GF(2) Matrix Inverter with Linear Time Complexity, Jasinski, R.P.; Pedroni, V.A.; Gortan, A.; Godoy, W.; Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on, 13-15 Dec. 2010 , 322 – 327, Quintana Roo
3. J.W. Nilsson, S.A. Riedel, Electric Circuits, Prentice Hall, New Jersey, 2011, pp. 705.
4. Y. Zhuo, X-L Wu, J.P. Haldar, W-M. Hwu, Z-P Liang, B.P. Sutton, Accelerating iterative field-compensated MR image reconstruction on GPUs, Biomedical Imaging: From Nano to Macro, 2010 IEEE International Symposium on,14-17 April 2010, 820 – 823.
5. Retrieved June 24, 2011 from the World Wide Web: http://developer.download.nvidia.com/compute/cuda/2_1/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.1.pdf.
6. Retrieved May 25, 2011 from the World Wide Web: http://xcr.cenit.latech.edu/hapc/docs/cuda.pdf.
7. X. Xia ,J.C. Lee, CUDA Linear Equations Solver Based on Modified Gaussian Elimination, Arizona University.
8. Rastegarpour, H. (2011). What Is The Hoopla About Blended Learning: Something Old Is New Again. *World Journal on Educational Technology, 3*(1), 39-47.