



Ain Shams University
Faculty of Engineering

Computers and Systems Department

Graph Based Techniques Applied in Electrical Circuit Simulation

A Thesis

Submitted in partial fulfillment for the requirements of
Doctorate of Philosophy of Science degree in Electrical
Engineering

Submitted by:

Hazem Said Ahmed Mohammed

M.Sc. of Electrical Engineering

(Computers and Systems Department)

Ain Shams University, 2006.

Supervised by:

Prof. Dr. Hussein Ismail Shahein

Prof. Dr. Hazem Mahmoud Abbas

Cairo 2014

Acknowledgements

All praise is due to Allah, Most Merciful, the Lord of the Worlds, Who taught man what he knew not. I would like to thank God Almighty for bestowing upon me the chance, strength and ability to complete this work.

I wish to express my gratitude to my supervisors, Professor Hussein Shahein and Professor Hazem Abbas for their exceptional guidance, encouragement, insightful thoughts and useful discussions. It was my great pleasure to work with Professor Hussein Shahein and Professor Hazem Abbas.

I would also like to extend my sincere appreciation to Dr. Bassem Amin and Dr. Watheq ElKharashi for their advice and support during my research.

I am in no way capable of appropriately thanking my parents and my wife for their unconditional love and unlimited support.

Hazem Said Ahmed
Computer and Systems Department
Faculty of Engineering
Ain Shams University
Cairo, Egypt
2013

Abstract

In this dissertation, the acceleration of electrical circuit simulation problem is addressed. As SPICE simulator core engine is the base of most current electrical circuit simulator, this engine is studied for detecting performance bottlenecks. Due its time complexity, solving sparse matrix step in SPICE engine is targeted in this dissertation to enhance the performance of the simulator.

A novel technique based on graph theory is introduced to solve sparse linear systems. The proposed technique enhances the ability to build sparse parallel solvers for circuit simulation matrices. Thus the proposed technique accelerated the performance of circuit simulation algorithms. The new technique represents sparse linear system as a signal flow graph (SFG). Then it divides the graph into separate strongly connected components (SCCs). SCCs relations are detected and represented in reduced graphs which are used to enhance the parallelism of the solver. To benefit from the parallel nature of the reduced graph representation, Graphics Processing Unit (GPU) is used to accelerate sparse Lower Upper (LU) factorization.

The main contribution in this dissertation is the parallelization of KLU ("Clark Kent" LU) algorithm through introducing the concept of the reduced graph. A theory that states that Reduced Graph are Non-Cyclic and hence can be processed in parallel is introduced and proved. The GPU is used to implement proposed parallel KLU. To validate the performance of the proposed technique, it is tested against real circuit matrices from the University of Florida sparse matrix collection. The proposed technique outperformed sequential KLU in most of the test cases.

Statement

This dissertation is submitted to Ain Shams University for the degree of Doctor of Philosophy in Electrical Engineering (Computers and Systems Engineering).

The work included in this thesis was carried out by the author at the Computers and Systems Engineering Department, Faculty of Engineering, Ain Shams University.

No part of this thesis has been submitted for a degree or qualification at other university or institution.

Hazem Said Ahmed Mohamed
Computers and Systems Engineering Department
Faculty of Engineering
Ain Shams University
Cairo, Egypt
2014

Contents

List of tables	viii
List of figures	ix
List of Abbreviations	xiii
1 Introduction	1
1.1 Problem Statement	2
1.2 Motivation	3
1.3 Dissertation Organization	3
2 Background	5
2.1 SPICE simulator	5
2.1.1 History of SPICE	5
2.1.2 SPICE Algorithm	6
2.1.2.1 Circuit Example	8
2.2 Sparse Linear Solvers	17
2.2.1 Dense LU	18
2.2.1.1 Gaussian Elimination	18

2.2.1.2	LU Factorization	19
2.2.2	Sparse LU	21
2.2.2.1	Sparse Matrix Representation	21
2.2.2.2	Sparse Triangular Solution	22
2.2.2.3	Gilbert/Peierls' Algorithm	26
2.2.2.3.1	Left Looking LU factorization	26
2.2.2.3.2	Calculating vector u_{12} :	26
2.2.2.3.3	Calculating element u_{22} :	27
2.2.2.3.4	Calculating vector l_{32} :	27
2.2.2.4	Gilbert/Peierls' Algorithm steps:	28
2.2.3	Fill-in	29
2.2.4	Ordering	31
2.2.4.1	Ordering effect on performance	31
2.2.4.2	Ordering methods	35
2.2.5	Factorization Methods	35
2.2.5.1	SuperLU	36
2.2.5.1.1	Super-nodes	36
2.2.5.2	KLU	38
2.2.5.2.1	Diagonal Free Matrix	39
2.2.5.2.2	BTF	40
2.2.5.2.3	Ordering	40
2.2.5.2.4	Block Factorization	40
2.2.5.2.5	Block Back Substitution	41
2.2.5.3	Accelerating KLU	42
3	Graph Based Parallel Sparse Solver	43
3.1	KLU steps	44

3.2	Proposed Technique	47
3.2.1	Zero Free Diagonal	49
3.2.2	Block Triangular Matrix	50
3.2.3	Ordering Diagonal Blocks	53
3.2.4	Factorizing Diagonal Blocks	54
3.2.5	Solving the System	54
3.2.6	Reduced Graph	54
3.3	Reduced Graph Processing Steps	56
3.3.1	Node Processing	56
3.3.2	Link Processing	57
3.4	Advantages of Proposed Technique	57
3.4.1	Inherited Parallelism	57
3.4.2	Memory Access	58
3.5	Next Steps	58
4	Implementation on GPU	59
4.1	GPU architecture	59
4.1.1	Streaming Multiprocessor	61
4.1.2	Compute Unified Device Architecture CUDA	61
4.1.3	CUDA Parallelism	63
4.2	GPU Based KLU technique	65
4.2.1	Input Matrix	66
4.2.2	Reduced Graph Representation	69
4.2.3	Nodes Processing Steps	70
4.2.3.1	Triangular Solution	71
4.2.3.2	Basic floating point step of Node Processing	72
4.2.4	Link Processing Steps	73

4.2.4.1	Basic floating point step of Link Processing	74
4.2.5	Planning Independent Processing Steps	74
4.2.6	SIMT compatibility	76
5	Experimental Results	79
5.1	University of Florida Sparse Matrix Collection	79
5.2	Test Platform	80
5.3	Algorithm Performance	80
5.4	Results Analysis	83
5.4.1	Parallel GPU KLU vs KLU	83
5.4.2	Parallel GPU KLU vs SuperLU and KLU	84
6	Conclusions	87
6.1	Findings	87
6.2	Future Work	88
	References	89

List of Tables

5.1	Test Matrices Statistics	81
5.2	Algorithm Performance Comparison on test Matrices	82

List of Figures

2.1	SPICE flowchart	7
2.2	Circuit Example	8
2.3	Capacitor equivalent circuit	12
2.4	Diode model linearization	13
2.5	Diode equivalent circuit	14
2.6	Graph representing relations between variable	24
2.7	Non-zero pattern of lower triangular system solution	25
2.8	Gilbert/Peierls' Algorithm matrix processing	29
2.9	Types of super-nodes	36
3.1	KLU steps	46
3.2	Graph representing the equations 3.6	49
3.3	Graph representing zero free diagonal matrix	51
3.4	SCC detection	52
3.5	Reduced Graph	55
3.6	Reduced Graph 2	55
4.1	GPU Architecture (figure from [42])	60
4.2	CUDA Programming Model (figure from [42])	62
4.3	Sample Matrix A	66

4.4	Sample Matrix Blocks	67
4.5	Reduced Graph of Sample Matrix	70
4.6	Node A_{44} processing steps	71
4.7	Link A_{34} Processing	73
4.8	Reduced Graph Processing Steps	75
5.1	Performance of KLU vs Parallel GPU KLU	83
5.2	Performance of KLU, SuperLU, and Parallel GPU KLU . . .	85

List of Abbreviations

AMD	Approximate Minimum Degree
BLAS	Basic Linear Algebra Subprograms
BTF	Block Triangular Form
COLAMD	Column Approximate Minimum Degree
CUDA	Compute Unified Device Architecture
FPGA	Field Programmable Gate Array
GPU	Graphical Processing Unit
IC	Integrated Circuit
KCL	Kirchhoff's Current Law
KLU	Clark Kent Lower Upper
LU	Lower Upper
MNA	Modified Nodal Analysis
SCC	Strongly Connected Component
SFG	Signal Flow Graph
SIMD	Single Input Multiple Data
SIMT	Single Input Multiple Threads
SM	Streaming Multiprocessors
SoC	System on Chip
SP	Streaming Processor

SPICE Simulation Program with Integrated Circuit Emphasis