View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by University of Thessaly Institutional Repository

Algorithms for Large-Scale Power Delivery Network Analysis on Massively Parallel Architectures

by

Konstantis Daloukas

Submitted to the Department of Electrical and Computer Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

at the

UNIVERSITY OF THESSALY

July 2014

©Konstantis Daloukas, 2014.

Author .....
Konstantis Daloukas
Department of Electrical and Computer Engineering
July 4, 2014

Certified by .....
Panagiota Tsompanopoulou
Assistant Professor
Thesis Supervisor

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Algorithms for Large-Scale Power Delivery Network Analysis on Massively Parallel Architectures

by

Konstantis Daloukas

Submitted to the Department of Electrical and Computer Engineering
on July 4, 2014, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical and Computer Engineering

Abstract

The on-chip power delivery network constitutes a vital subsystem of modern nanometer-scale integrated circuits, since it affects in a critical way the performance and correct operation of the devices. As technology scaling enters in the nanometer regime, there is an increasing need for accurate and efficient analysis of the power delivery network. The impact of first-order phenomena like IR drop or electromigration or second-order phenomena like Joule heating, that were neglected until recently, on the power delivery network, necessitates the existence of fast and accurate methodologies for electrical and thermal analysis of the power grid.

A typical power delivery network is modeled as an RLC network and its electrical or thermal analysis amounts at solving a linear system of equations. Due to the sheer size of contemporary power delivery networks (which comprise millions or billions of nodes), its analysis is a very challenging process, both in terms of computational and memory requirements. Parallel architectures that have recently appeared and provide a large amount of computational resources appear as the platform of choice for executing computationally demanding algorithms. However, most state-of-the-art algorithms for power grid analysis do not entail a large degree of parallelism and have excessive memory requirements, which makes their mapping onto parallel architectures difficult or even infeasible.

To this end, this dissertation proposes three new methodologies for analysis of large-scale power delivery networks found in contemporary integrated circuits. We present two algorithms for electrical analysis and one algorithm for combined electro-thermal analysis of the power delivery network. The novel characteristic of the proposed algorithms is the large degree of multi-level parallelism that they entail. As a results, they appear as ideal candidates for mapping onto parallel architectures. Our algorithms are able to greatly accelerate the simulation process, achieving up to two or three orders of magnitude speedup for power grid electrical analysis and one order of magnitude speedup for electro-thermal analysis, while at the same time scaling linearly with the number of power grid nodes.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Αλγόριθμοι για Ανάλυση του Δικτύου Τροφοδοσίας Μεγάλης Κλίμακας Ολοκληρωμένων Κυκλωμάτων σε Μαζικά Παράλληλες Αρχιτεκτονικές

Κωνσταντής Νταλούκας

Περίληψη

Το δίκτυο τροφοδοσίας αποτελεί ένα ζωτικής σημασίας υποσύστημα στα μοντέρνα ολοκληρωμένα κυκλώματα, καθώς καθορίζει τη σωστή λειτουργία και την απόδοση του κυκλώματος. Λόγω της συνεχούς μείωσης της κλίμακας σχεδιασμού, υπάρχει μια αυξανόμενη ανάγκη για ακριβή και αποδοτική ανάλυση του δικτύου τροφοδοσίας. Ο αντίκτυπος των πρωτευόντων φαινομένων, όπως η πτώση τάσης ή η ηλεκτρομετανάστευση, αλλά και των δευτερευόντων φαινομένων, όπως η επίδραση του φαινομένου Joule heating στους αγωγούς, που δε λαμβάνονταν υπόψη μέχρι πρότινος, καθιστούν απαραίτητη την ύπαρξη κατάλληλων και αποδοτικών μεθοδολογιών για ηλεκτρική και θερμική ανάλυση του δικτύου τροφοδοσίας.

Ένα τυπικό δίκτυο τροφοδοσίας μοντελοποιείται ως ένα δίκτυο RLC και η ηλεκτρική και θερμική ανάλυσή του ανάγεται στην επίλυση ενός συστήματος γραμμικών εξισώσεων. Λόγω του μεγάλου μεγέθους των σύγχρονων δικτύων τροφοδοσίας (τα οποία περιλαμβάνουν εκατομμύρια ή δισεκατομμύρια κόμβους), η ανάλυσή τους είναι μια πολύ απαιτητική διαδικασία, τόσο από άποψη υπολογιστικής ισχύος όσο και από άποψη απαιτήσεων σε μνήμη. Οι παράλληλες αρχιτεκτονικές που έχουν εμφανιστεί πρόσφατα, προσφέρουν πολύ μεγάλη υπολογιστική ισχύ με αποτέλεσμα να εμφανίζονται ως η κατάλληλη πλατφόρμα για την εκτέλεση υπολογιστικά απαιτητικών αλγορίθμων. Ωστόσο, οι περισσότεροι αλγόριθμοι που χρησιμοποιούνται σήμερα στα περισσότερα εμπορικά εργαλεία για την ανάλυση του δικτύου τροφοδοσίας δεν εμπεριέχουν μεγάλο βαθμό παραλληλισμού και έχουν υπερβολικές απαιτήσεις μνήμης, γεγονός που καθιστά την υλοποίησή τους σε παράλληλες αρχιτεκτονικές ιδιαίτερα δύσκολη και σε αρκετές περιπτώσεις ανέφικτη.

Βάσει των παραπάνω, η παρούσα διατριβή προτείνει τρεις νέες μεθοδολογίες για την

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

ανάλυση των δικτύων τροφοδοσίας μεγάλης κλίμακας που εμπεριέχονται στα σύγχρονα ολοκληρωμένα κυκλώματα. Παρουσιάζονται δύο αλγόριθμοι για την ηλεκτρική ανάλυση και ένας αλγόριθμος για τη συνδυασμένη ηλεκτρο-θερμική ανάλυση του δικτύου τροφοδοσίας. Το νέο χαρακτηριστικό των προτεινόμενων αλγορίθμων είναι ο μεγάλος βαθμός παραλληλισμού σε πολλαπλά επίπεδα, ο οποίος τους καθιστά κατάλληλους για την υλοποίηση σε παράλληλες αρχιτεκτονικές. Οι προτεινόμενοι αλγόριθμοι επιταχύνουν σημαντικά τη διαδικασία προσομοίωσης, επιτυγχάνοντας έως και τρεις τάξεις μεγέθους επιτάχυνση για την ηλεκτρική ανάλυση και μία τάξη μεγέθους επιτάχυνση για την ηλεκτρο-θερμική ανάλυση, ενώ την ίδια στιγμή κλιμακώνουν γραμμικά με τον αριθμό των κόμβων στο δίκτυο τροφοδοσίας τόσο σε υπολογιστική πολυπλοκότητα όσο και σε απαιτήσεις σε μνήμη.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Acknowledgements

As this long endeavor has reached its end, I would like to express my deep gratitude to a group of people who provided my with guidance, assistance, support, and encouragement. It is certain that without their help, I would not have been able to complete my dissertation.

First of all, I would like to express my sincere gratitude to my Ph.D. advisors, Professor Panagiota Tsompanopoulou, Professor Nestor Evmorfopoulos, and Professor George Stamoulis. Their guidance and support were the main reasons that most ideas presented in this dissertation have been accomplished. Their diligence, intelligence and wisdom have had great influence on my research development. Research directions, technical criticisms as well as valuable feedback from our office discussions brought me continuous sources of motivation and inspiration. Apart from being excellent scientists, they are also valuable collaborators and true supporters.

I would also like to thank my thesis committee members, Professor John Moondanos and Professor Christos Sotiriou from the University of Thessaly, as well as Professor Apostolos Dollas and Professor Dionisios Pnevmatikatos from the Technical University of Crete, for providing important feedback for my dissertation.

My sincere thanks to all my friends for their encouragement all these years. Especially, I would like to thank my friends from the Electronics Lab. Without their help, I would not have led such a fruitful and joyful life in the Department of Electrical and Computer Engineering. Particularly, I would like to thank my close friends, George, Michalis, and Babis for their constant support, help and valuable discussions. Also, many thanks to Mary, Lena, Maria, and Despoina for their continuous secretarial support.

During the academic years 2010-2013, my research had been partially funded by the Bodossaki Foundation in Greece. I really appreciate the opportunity of being a member of the Bodossaki Foundation scholars' family and the experience of participating in a large number of conferences that made my research more visible.

Also, I would like to thank Yorgos Koutsoyannopoulos and Sotiris Bantas from

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Helic, Inc. for their guidance and financial support during the last year of my studies.

Finally, I appreciate so much the unconditional love, help, trust, and support from my family and Ioanna. There are no words that can express my gratitude and appreciation for all they have done for me. Without their patience and support, I would not have been able to finish my dissertation. The least I can do in recognition is to dedicate this dissertation to them.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

To my family, Ioanna & my friends

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

This doctoral thesis has been examined by a joint committee of the Department of Electrical and Computer Engineering from the University of Thessaly and the Department of Electronic and Computer Engineering from Technical University of Crete as follows:

Professor Panagiota Tsompanopoulou .....
Chairwoman, Thesis Supervisor
Assistant Professor of Electrical and Computer Engineering
University of Thessaly

Professor Nestor Evmorfopoulos .....
Member, Thesis Committee
Assistant Professor of Electrical and Computer Engineering
University of Thessaly

Professor George Stamoulis .....
Member, Thesis Committee
Professor of Electrical and Computer Engineering
University of Thessaly

Professor Apostolos Dollas .....
Member, Examination Committee
Professor of Electronic and Computer Engineering
Technical University of Crete

Professor John Moondanos .....
Member, Examination Committee
Professor of Electrical and Computer Engineering
University of Thessaly

Professor Dionisios Pnevmatikatos .....
Member, Examination Committee
Professor of Electronic and Computer Engineering
Technical University of Crete

Professor Christos Sotiriou .....
Member, Examination Committee
Professor of Electrical and Computer Engineering
University of Thessaly

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Contents

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

List of Figures

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

List of Tables

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

List of Acronyms

ALU	Arithmetic Logic Unit
BiCG	Biconjugate Gradient Method
CG	Conjugate Gradient
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DCT	Discrete Cosine Transform
FDM	Finite Difference Method
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Arrays
GMRES	Generalized Minimal Residual Method
GPU	Graphics Processing Unit
IC	Integrated Circuit
IDCT	Inverse Discrete Cosine Transform
MIMD	Multiple Instruction Multiple Data
MNA	Modified Nodal Analysis
ODE	Ordinary Differential Equation
PCG	Preconditioned Conjugate Gradient
SFU	Special Function Units
SIMD	Single Instruction Multiple Data
SIMT	Single Instruction Multiple Thread
SMX	Streaming Multiprocessor
SP	Streaming Processor

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

SPD Symmetric and Positive Definite

TSV Through Silicon Via

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Chapter 1

Introduction

1.1 Background and Motivation

The relentless push for high-performance and low-power integrated circuits has been met by aggressive technology scaling, which enabled the integration of a vast number of devices on the same die but brought new problems and challenges to the surface. The on-chip power delivery network (power grid) constitutes a vital subsystem of modern nanometer-scale Integrated Circuits (ICs), since it affects in a critical way the performance and correct operation of the devices.

The major issues in the power delivery networks of contemporary circuits is voltage drop (or IR drop) and electromigration. Voltage drop is the decrease in the nominal value of the supply voltage of the chip that a device sees due to the finite resistance of interconnects and can severely affect the correct operation and the performance of the design. On the other hand, electromigration is the transport of material caused by the gradual movement of the ions in a conductor due to the momentum transfer between conducting electrons and diffusing metal atoms. Electromigration affects the reliability of the power grid and can even lead to disconnections, thus making a chip unusable.

As a result, in order to address these issues and to determine the quality of the supply voltage delivered to the devices, the designer has to perform static and dynamic simulation of the electrical circuit modeling the power grid. This

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

has become a very challenging problem for contemporary ICs, since power grids encountered in these circuits are extremely large (comprising several thousands or millions of nodes) and very difficult to simulate efficiently (especially over multiple time-steps). Static (DC) or transient simulation refers to the process of computing the response of an electrical circuit to a constant or time-varying stimulus respectively. Since a power delivery network can be generally modeled as a linear RLC circuit, the process of DC or transient simulation of large-scale power grids amounts to solving very large (and sparse) linear systems of equations. The methods that have been proposed so far for tackling analysis of power delivery networks can be categorized as follows:

- Direct methods (based on matrix factorization) have been widely used in the past for solving the resulting linear systems arising in power grid analysis, mainly because of their robustness in most types of problems. They also have the property of reusability of factorization results in transient simulation with a fixed time-step. Unfortunately, these methods do not scale well with the dimension of the linear system, and become prohibitively expensive for circuits beyond a few thousand elements, in both execution time and memory requirements. In addition, a fixed time-step is almost never used in practice because it becomes very inefficient to constantly simulate during long intervals of low activity. All practical implementations of integration techniques for Ordinary Differential Equations (ODEs) employ a variable or adaptive time-step mechanism [9]. In those cases, the reusability of matrix factorization in direct methods ceases to exist.

- Iterative methods involve only inner products and matrix-vector products, and constitute a better alternative for large sparse linear systems in many respects, being more computationally- and memory-efficient. This holds even more so for modern nonstationary iterative methods which fall under the broad class of "Krylov-subspace" methods [31]. Iterative methods possess themselves a kind of reusability property for transient simulation, in that the solution at the

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

last time-step provides an excellent initial guess for the next time-step, thus making a properly implemented iterative method converge in a fairly small number of iterations. In fact, this property also holds in the case of a variable time-step, since the quality of the last solution as initial guess for the next solution is not affected. The above features make iterative methods much more suitable for DC and variable time-step transient analysis of large-scale linear circuits such as power distribution networks.

The main problem of iterative methods is their unpredictable rate of convergence which depends greatly on the properties (specifically the condition number) of the system matrix. A preconditioning mechanism, which transforms the linear system into one with more favorable properties, is essential to guarantee fast and robust convergence. However, the ideal preconditioner (one that approximates the system matrix well and is inexpensive to construct and apply) differs according to each particular problem and each different type of system matrix. That is why iterative methods have not reached the maturity of direct methods and have not yet gained widespread acceptance in linear circuit simulation. Although general-purpose preconditioners (such as incomplete factorizations or sparse approximate inverses) have been developed, they are not tuned to any particular simulation problems and cannot improve convergence by as much as specially-tailored preconditioners.

Another aspect of circuit simulation that has become very important recently is to uncover hidden opportunities for parallelism in its intermediate steps. This is essential for harnessing the potential of contemporary parallel architectures, such as multi-core processors and Graphics Processing Units (GPUs) in order to enable analysis for very-large scale power delivery networks. GPUs, in particular, are massively parallel architectures whose computational power is about 3.95 TFlops, greater by an order of magnitude than that of multi-core processors. As a result, they appear as a platform of choice for the efficient execution of computationally-intensive tasks such as power grid analysis and simulation.

Direct solution methods offer little room for parallelism as they are mainly based on backward and forward solution of triangular systems which do not entail large

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

parallelism due to the dependencies that exist among the solution steps. On the contrary, Krylov-subspace iterative methods offer ample possibilities for parallelism that have been explored sufficiently well. However, the construction and application of the preconditioner is a very delicate part of parallelizing an iterative method because it is completely application-dependent (and traditional general-purpose preconditioners have very little room for parallelism). Unfortunately, there has been little systematic research for the development of parallel simulation algorithms, and more specifically algorithms for power grid analysis that can be mapped onto massively parallel architectures like GPUs. This can be attributed in part to the difficulty in parallelization of direct linear solution methods that have been mostly employed thus far. In addition, most sophisticated preconditioners that have been developed for power grid analysis have little room for parallelism. Thus, they cannot be mapped efficiently and take full advantage of the computational power of massively parallel architectures.

1.2 Contributions

In order to address the limitations of the existing simulation techniques for power grid analysis, this research work presents a class of new parallel algorithms for efficient analysis of power delivery networks found in contemporary large-scale ICs as described below:

1. Two parallel algorithms for DC and transient electrical analysis of power delivery networks, FTCG and FTCG-3D. FTCG targets power delivery networks with negligible via resistances (near-2D structures), while FTCG-3D targets power delivery networks with significant via resistances. Both methods combine a preconditioned iterative method with two problem-specific and highly-parallel preconditioning algorithms. Both preconditioning algorithms take into account the structure of the underlying power grid in order to accelerate the convergence of the iterative method. In addition, their specialized structure allows applying a Fast Transform-based solver that utilizes Fast

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Fourier Transform (FFT) for the solution of the necessary preconditioning step. The main characteristics of the application of a Fast Transform are the near-optimal operation complexity, as well as its inherent parallelism and low memory requirements, compared to a generic solver for linear systems. As a result, massively parallel architectures such as GPUs can be used to accelerate the simulation algorithm, while at the same time the application's memory demands render feasible the analysis of very large power grids on such architectures.

2. A parallel algorithm (ET-FTCG) for static combined electro-thermal analysis of power delivery networks. ET-FTCG combines FTCG for electrical analysis and an efficient preconditioning approach for thermal analysis and can efficiently tackle electro-thermal analysis of very large-scale power delivery networks by utilizing massively parallel architectures.

1.3 Outline

The next chapters of the dissertation are organized as follows. Firstly, the relevant background information regarding parallel architectures are presented in Chapter 2. Then, in Chapter 3 we give the important background details behind direct and iterative linear system solution methods, preconditioning and porting of linear system solution algorithms onto parallel architectures. Chapter 4 describes the theory behind power grid electrical analysis and discusses the proposed algorithms for large-scale power grid analysis on massively parallel architectures. Following this chapter, we describe the details behind our proposed algorithm for electro-thermal analysis of the power delivery network in Chapter 5. The proposed algorithm combines one of the techniques described in Chapter 4 with a novel algorithm for thermal analysis of the power grid. Finally, Chapter 6 concludes the dissertation.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Chapter 2

Parallel Computing Architectures

2.1 Introduction

During the last few years, we have witnessed a paradigm shift towards parallel computing. Technology advances have enabled the integration of multiple cores in a single die, thus enabling the development of a plethora of computation substrates for parallel, high performance, computing. In addition, the diminishing returns from the continuous technology scaling on single-core processors has forced researchers to start designing architectures that would incorporate more than one processors. Architectures such as homogeneous or heterogeneous multicore and manycore processors, and, more recently, GPUs have allowed the time- and power efficient execution of computationally intensive applications at a minimum expense.

Multi-core processors seek to maintain the execution speed of sequential programs as more cores are utilized. Starting from a small number of processors and owing to technology scaling, designers increased the number of cores at each semiconductor process generation. Each core comprises its own local cache and computational resources, while communication between different cores is achieved through means of high-speed interconnects. Although multi-core processors offer a significant amount of computational resources, they are mainly optimized for execution of sequential applications. As a result, porting of an application on a multi-core architecture is not a trivial process.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Figure 2-1: Architecture of the NVIDIA Tesla K20 GPU.

In contrast to multi-core architectures, massively parallel architectures (and their main representatives GPUs) comprise a large number of simpler computational cores, as they are focused on providing massive parallelism.

Fig. 2.1 depicts the architecture of the latest Tesla K20 GPUs accelerator from NVIDIA, which is a typical high-end GPUs. The smallest unit capable of performing parallel computations is the Streaming Multiprocessor (SMX), with the main difference between a low-end GPUs and a high-end GPUs of the same architecture being the number of SMXs inside the chip. In the case of Tesla K20 GPUs, each SMX unit is composed of 192 cores, also known as Streaming Processors (SPs). Its architecture was built for a maximum of 15 SMXs, giving a maximum of 2,880 cores. However in practice, some SMXs are deactivated in order to increase the yield.

The cores of a SMX are mainly optimized for calculations which means that there is no extra hardware devoted to sequential and control operations (e.g. branch prediction). They are 32-bit units that can perform basic integer and single precision (FP32) floating point arithmetic. In addition to the computation cores, there are 32 Special Function Unitss (SFUs) that perform special mathematical operations such as log, sqrt, sin and cos, among others. Each SMX has also 64 double precision

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

floating point units, known as FP64, and 32 LD/ST units (load / store) for writing and reading memory.

As far as the memory model of the GPU is concerned, GPUs such as the Tesla K20 implement a four-level memory hierarchy; (1)registers, (2) L1 cache, (3) L2 cache and (4) global memory. All levels, except for the global memory, reside in the GPU chip. The L2 cache is hardware-managed and it improves memory accesses on global memory. The L1 cache is software-managed (meaning that the programmer is responsible for managing its contents), there is one per SMX, and it can be as fast as the registers. Kepler and Fermi based GPUs have L1 caches of size 64KB that are split into 16KB of programmable shared memory and 48KB of automatic cache, or vice versa.

2.2 Differences between CPUs and GPUs

Modern GPUs have evolved towards parallel processing, implementing the Multiple Instruction Multiple Data (MIMD) architecture. Most of their chip budget is reserved for control units and cache, leaving a small area for numerical computations. A Central Processing Unit (CPU) performs different tasks and advanced control and cache mechanisms is the only way to achieve a good performance level. On the other hand, GPUs have a Single Instruction Multiple Data (SIMD) architecture and the main goal of its architecture is to achieve high performance through massive parallelism. Contrary to the CPU, a GPU is mostly occupied by Arithmetic Logic Units (ALUs) and a minimal region is reserved for control and cache. Owing to their architecture, GPUs can achieve up to three orders of magnitude speedup over CPUs for algorithms that entail large degree of parallelism.

Although this difference in architecture makes GPUs much more restrictive than CPUs, a GPU is much more powerful if an algorithm is carefully designed for it. Contemporary GPU architectures such as NVIDIA's Fermi and Kepler have added a significant degree of flexibility by incorporating a L2 cache for handling irregular memory accesses and by improving the performance of atomic operations, even if

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

such flexibility is still far from the one found in CPUs. As a result, there is a trade-off between flexibility and computing power. Actual CPUs struggle to maintain a balance between computing power and general purpose functionality. On the other hand, GPUs aim at massive parallel arithmetic computations.

2.2.1 Concurrency in GPUs

Usually, the number of logical threads that a GPU application requires is larger than the available processing units in the GPU hardware. As a result, there must be an efficient way for thread management that will allow for efficient utilization of the GPU resources.

GPUs divide the number of threads into small groups that work in SIMD mode. For the NVIDIA GPUs, these groups are known as warps and each warp contains 32 threads. GPUs comprise efficient mechanisms for handling the entire space of computation in order to support concurrency. The number of threads that are running on the GPU corresponds to the number of processing units available. However, the maximum number of concurrent threads available is much higher. For example, the latest Tesla K40 GPU can process up to 2,880 threads in parallel, but can handle up to 30,720 concurrent threads. The thread scheduler is the hardware part that is responsible for deciding which warp of threads is ready for execution. It switches idle warps (e.g., warps that are waiting for a memory access) with warps ready for computation. This means that numerical computations and memory accesses are pipelined and the thread scheduler tries to maintain this pipeline full all the time and fully utilize the computational resources of the GPU.

2.3 GPU Programming - The CUDA Programming Model

With the advent of more sophisticated GPUs, the need for programmability became apparent in order to utilize their computation resources. The Compute Unified Device Architecture (CUDA) [2] programming model is a parallel computing

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

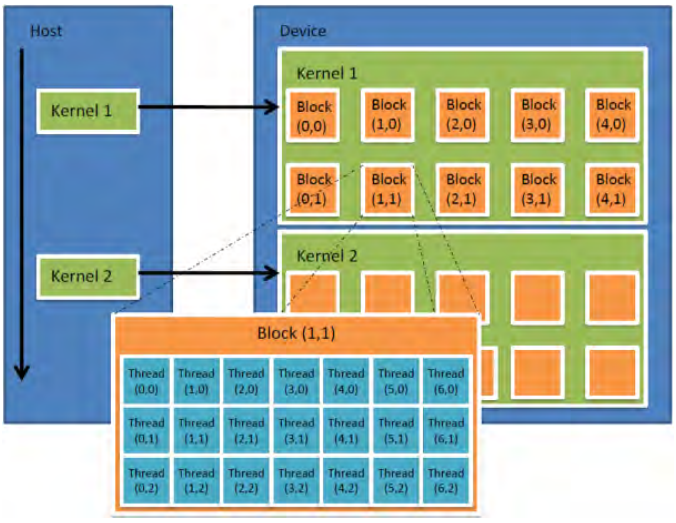


Figure 2-2: NVIDIA CUDA Thread Model.

platform and programming model created by NVIDIA and implemented by the GPUs that they produce. An application written in the CUDA programming model comprises the host and the device code. The host code executes on the CPU-side and contains the control-intensive part of the application. It is responsible for uploading the data to the GPU and orchestrating the execution of the device code. The host program can work in a synchronous or asynchronous manner, depending if the result from the GPU is needed for the next step of computation or not. When the device code has finished in the GPU, the result data is copied back from device to host.

On the other hand, the device code comprises a series of kernel functions (or kernels) that execute on the GPU and correspond to the computational intensive parts of the application. CUDA programmers typically use kernels for expressing parallelism at its finest granularity. Through the abstraction layer that is offered by the CUDA programming model, the programmer can design massively parallel algorithms independent of the number of physical processing units or the execution order of threads.

Upon execution of a kernel function, the programmer defines a logical geometry. This ”geometry” of execution is described by a 3-level, 3D index space where all logical threads are organized. The space of computation is composed of a grid,

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

blocks and threads. An example of a two dimensional grid of computation depicted in Fig. 2-2. A grid is a discrete k-dimensional (with k = 1,2,3) box type structure that defines the size and volume of the space of computation. Each element of the grid is a block and each block contains many spatially organized threads. A thread is assigned to every point in the k-dimensional space and corresponds to the execution of a particular instance of the kernel function. Each thread is described by a unique tuple of ids. Threads are organized into blocks, each having up to three dimensions (3D thread index within the block geometry). The overall computation can, in turn, be partitioned in blocks, also organized in a 3D space (3D block index within the global computation geometry). CUDA provides functionality for synchronization among threads that belong to the same block. On the other hand, blocks are completely independent on each other and can execute in parallel. Therefore, only threads that belong to the same block can communicate directly, through memory which is visible only inside the block.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Chapter 3

Linear System Solution Methods

In general, a linear system of equations is described by the following equation:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.1}$$

where $\mathbf{A} \in \Re^{n \times n}$ is an $n \times n$ matrix of real numbers, $\mathbf{b} \in \Re^n$ is a vector of size n, and $\mathbf{x} \in \Re^n$ is an unknown solution of vector of size n that will be determined by a solution method. A solution for the linear system in (3.1) exists if the matrix $\mathbf{A}$ is non-singular, which means that the inverse matrix $\mathbf{A}^{-1}$ with $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$ exists. The solution methods for linear systems are classified as **direct** and **iterative**. Direct solution methods solve the above linear system in a predefined number of steps, which depends on the size n of the linear system. On the other hand, iterative solution methods determine an approximation of the exact solution to a predefined accuracy level.

Matrices arising in power grid analysis are very large (n can be in the order of millions or billions) and have two important features:

- The system matrix $\mathbf{A}$ is sparse, with less than 20 elements per each row, in principal. Had the system matrix been full, power grid analysis would be impractical due to both execution time and memory reasons.

- System matrix $\mathbf{A}$ is a Symmetric and Positive Definite (SPD) matrix. Symmetry

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

means $\alpha_{ij} = \alpha_{ji},\ i, j = 0, 1, \ldots, n-1$, while positive definiteness means that

$$\mathbf{v}^T \mathbf{A}\mathbf{v} > 0,\ for\ all\ vectors\ \mathbf{v}$$

Symmetry and positive definiteness are two important properties of a matrix that allow utilization of more efficient direct or iterative methods for the solution of the corresponding system.

This chapter describes the general principles that sparse direct and iterative methods are based on and describes the state-of-the-art algorithms for solution of symmetric and positive-definite linear system of equations, namely the Cholesky decomposition and the Preconditioned Conjugate Gradients algorithms.

3.1 Direct Methods

Direct solution methods solve the linear system in (3.1) in a predefined number of steps, which depends on the size of the linear system. They consist of two steps, namely a factorization step where the system matrix is decomposed into a number of factors, and the solution phase where the matrix factorization is used for the solution of the initial system.

LU factorization is the direct method used in general, non-symmetric matrices. It factors the system matrix in two factors, one lower- and one upper-triangular matrix $\mathbf{A} = \mathbf{LU}$, and equation (3.1) is transformed into the following:

$$\mathbf{Ax} = \mathbf{b} \Rightarrow (\mathbf{LU})\mathbf{x} = \mathbf{b} \Rightarrow \mathbf{L}(\mathbf{Ux}) = \mathbf{b}$$

where $\mathbf{L}$ is a lower-triangular matrix while $\mathbf{U}$ is an upper-triangular one. As a result, the original system is transformed into two equivalent systems and is solved into two steps as follows:

$$\begin{aligned} \mathbf{Ly} &= \mathbf{b} \\ \mathbf{Ux} &= \mathbf{y} \end{aligned}$$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Algorithm 1 Factorization phase for the LU algorithm

```
for k = 0 : n − 2 do
  for i = k : n − 1 do
    a(i,k) = a(i,k) / a(k,k)
    for j = k + 1 : n − 1 do
      a(i,j) = a(i,j) - a(i,k) · a(k,j)
    end for
  end for
end for
```

The advantage of breaking the original linear system into the above set is that each linear system requires the solution of a triangular system (forward substitution for the first system and backward substitution for the second one) which is a trivial computational process.

Algorithm 1 presents the pseudocode for the factorization phase of the LU algorithm, where the **L** and **U** factors are stored in place, while Algorithm 2 and Algorithm 3 present the pseudocode for the forward and backward substitution for the solution of the triangular systems. As we can observe, the factorization phase of the LU algorithm requires $\frac{2n^3}{3}$ operations, rendering the decomposition phase a computationally demanding process with the increasing size of the system matrix.

On the other hand, if matrix **A** is SPD, it allows for a special factorization and the Cholesky decomposition can be employed. Cholesky decomposition factors the system matrix into $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ and the original system is transformed into the

Algorithm 2 Forward substitution for solution of a lower triangular system

```
for i = 0 : n − 1 do
  y(i) = b(i);
  for j = 0 : i − 1 do
    y(i) -= L(i, j) * y(j);
  end for
  y(i) /= L(i, i);
end for
```


Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Algorithm 3 Backward substitution for solution of an upper triangular system

```
for i = n − 1 : 0 do
  x(i) = y(i);
  for j = i + 1 : n − 1 do
    x(i) -= U(i, j) * x(j);
  end for
  x(i) /= U(i, i);
end for
```

Algorithm 4 Factorization phase for the Cholesky algorithm

```
for i = 1 : n − 1 do
  for j = 0 : i do
    s = 0
    for k = 0 : j − 1 do
      s += L[i * n + k] * L[j * n + k];
      if i == j then
        L[i * n + j] = sqrt(A[i * n + i] - s)
      else
        L[i * n + j] = (1.0 / L[j * n + j] * (A[i * n + j] - s));
      end if
    end for
  end for
end for
```

following equivalent systems:

$$\begin{aligned} \mathbf{Ly} &= \mathbf{b} \\ \mathbf{L}^T\mathbf{x} &= \mathbf{y} \end{aligned}$$

Algorithm 4 presents the pseudocode for the decomposition phase of Cholesky factorization. Again, after matrix factorization, the resulting linear systems are solved by employing Algorithm 2 and Algorithm 3. As we can observe, Cholesky factorization is more efficient than LU factorization, requiring $\frac{n^3}{3}$ operations and half the amount of memory storage for saving the matrix factors.

The main advantages of direct methods are their robustness and the fact that once the factorization is completed, the solution of the linear system, even with multiple right hand side vectors, is a trivial process as long as the system matrix remains the same. However, direct methods present superlinear scaling both in

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

computational and memory requirements with the increasing size of the linear system, which rules them out for large-scale linear systems.

3.2 Iterative Methods

Iterative methods belong to the general category of relaxation methods. Starting with an initial solution guess, they provide a partial solution in each step which eventually converge to the desired solution, with a predefined accuracy level. Iterative methods can be categorized as follows:

- Stationary methods that solve a linear system with a matrix that approximates the original one through a series of steps that try to minimize the error of the result. The approximation matrix is usually a decomposition of the initial matrix that allows for more efficient solution. Among the most well-known stationary iterative methods is Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR). If $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$ is a decomposition of the system matrix in its diagonal ($\mathbf{D}$), upper triangular ($\mathbf{U}$), and lower triangular ($\mathbf{L}$) parts, the approximation for each of the aforementioned iterative method is presented below:

  - Jacobi: $\mathbf{x}^{(k+1)} = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{R}\mathbf{x}^{(k)})$, where $\mathbf{R} = \mathbf{U} + \mathbf{L}$.
  - Gauss-Seidel: $\mathbf{x}^{(k+1)} = \mathbf{L}_*^{-1}(\mathbf{b} - \mathbf{U}\mathbf{x}^{(k)})$, where $\mathbf{L}_* = \mathbf{L} + \mathbf{D}$.
  - SOR: $\mathbf{x}^{(k+1)} = (\mathbf{D}+\omega\mathbf{L})^{-1}\big(\omega\mathbf{b} - (\omega\mathbf{U} + (\omega-1)\mathbf{D})\big)\mathbf{x}^{(k)} = \mathbf{L}_w\mathbf{x}^{(k)} + \mathbf{c}$, where $\mathbf{L}_w = -(\mathbf{D} + \omega\mathbf{L})^{-1}(\omega\mathbf{U} + (\omega - 1)\mathbf{D})$, $\mathbf{c} = (\mathbf{D} + \omega\mathbf{L})^{-1}\omega\mathbf{b}$, and $0 < \omega < 2$.

- Non-stationary (or Krylov-subspace) methods: They form a basis of the sequence of successive matrix powers times the initial residual, which is called the Krylov sequence. Then, the approximations to the solution are formed by minimizing the residual over the subspace formed. Typical examples of Krylov-subspace methods are the Conjugate Gradient (CG), the Generalized Minimal Residual Method (GMRES), and the Biconjugate Gradient Method (BiCG). The CG

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

method is applicable to SPD systems while GMRES and BiCG are applicable on non-symmetric problems.

We are mainly interested in SPD matrices as this is the type of matrices involved in linear systems arising from power grid analysis. In the next section we will provide a detailed description of the CG iterative method that is applicable on SPD linear systems.

3.2.1 Conjugate Gradient Algorithm

The CG method is the first Krylov-subspace iterative method that was developed for SPD matrices [31]. The idea behind the CG algorithm is based on the theory of global minimization and orthogonal polynomials. Through a number of iterations, the CG method aims to minimize the A-norm:

$$\| x_i - x \|_{\mathbf{A}}^2 \equiv (x_i - x, \mathbf{A}(x_i - x))$$

for x_i that are in the Krylov subspace $K_i(\mathbf{A}, r_0) \equiv \{r_0, \ldots, \mathbf{A}^{i-1} r_0\}$. The CG method approximates the solution of the linear system by computing a series of residual vectors, where in each step the current residual vector is orthogonal to the space of the previously generated residuals. At its final iteration, the solution vector approximates the real solution of the initial system in a predefined accuracy level.

Regarding the convergence rate of CG, it can be shown [4] that the required number of iterations (for a given initial guess and convergence tolerance) is bounded in terms of the spectral condition number $\kappa_2(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2 \geq 1$ - specifically, it is $\mathcal{O}(\sqrt{\kappa_2(\mathbf{A})})$, which for SPD matrices becomes $\kappa_2(\mathbf{A}) = \frac{\lambda_{max}(\mathbf{A})}{\lambda_{min}(\mathbf{A})}$ where $\lambda_{max}(\mathbf{A})$, $\lambda_{min}(\mathbf{A})$ are the maximum and minimum eigenvalues of $\mathbf{A}$ respectively. This means that convergence of CG is fast when $\kappa_2(\mathbf{A}) \simeq 1$ and slow when $\kappa_2(\mathbf{A}) \gg 1$.

The main drawback of iterative methods is the unknown number of steps that are required for convergence (convergence rate). The convergence rate of iterative methods depends on the spectral properties of the matrix $\mathbf{A}$ of the linear system. In order to improve these properties, a mechanism that transforms the original matrix

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

to a matrix with more favorable properties is required. This process is known as preconditioning.

3.2.2 Preconditioning

Preconditioning is a technique that is used to transform the original linear system into one with more favorable spectral properties, and is essential to guarantee fast and robust convergence of an iterative method. In the case of linear systems involving SPD matrices, the rate of convergence of the conjugate gradient method depends on the distribution of the eigenvalues of $\mathbf{A}$. Hopefully, the transformed (preconditioned) matrix will have a smaller spectral condition number, and/or eigenvalues clustered around 1. Nevertheless, a clustered spectrum (away from 0) often results in rapid convergence, particularly when the preconditioned matrix is close to normal. If $\mathbf{M}$ denotes the preconditioner matrix, then the following linear system (left preconditioned system) has the same solution with the system from (3.1) but is easier to solve:

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.2}$$

In the case of Krylov subspace methods like the CG algorithm, it is not necessary to form the preconditioned matrix $\mathbf{M}^{-1}\mathbf{A}$ explicitly, as it would be too expensive and we would lose the sparsity of the matrix. Instead, matrix-vector products are required and a series of linear system solutions of the form:

$$\mathbf{M}\mathbf{z} = \mathbf{r} \tag{3.3}$$

Algorithm 5 describes the Preconditioned Conjugate Gradient (PCG) method for the solution of an SPD linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. As we can observe, PCG entails a large degree of parallelism as it comprises only matrix-vector and vector-vector products that make the method an ideal candidate for mapping onto parallel architectures. The preconditioner solve step $\mathbf{M}\mathbf{z} = \mathbf{r}$ in every iteration (line 6) effectively modifies the CG algorithm to solve the system $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$, which has the same solution as the original one $\mathbf{A}\mathbf{x} = \mathbf{b}$ [4]. In this case, the computationally demanding part of

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Algorithm 5 Preconditioned Conjugate Gradients

1: $\mathbf{x}$ = initial guess $\mathbf{x}^{(0)}$
2: $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$
3: $iter = 0$
4: **repeat**
5: $iter = iter + 1$
6: **Solve** $\mathbf{Mz} = \mathbf{r}$ (**Preconditioner Solve Step**)
7: $\rho = \mathbf{r} \cdot \mathbf{z}$
8: **if** $iter == 1$ **then**
9: $\mathbf{p} = \mathbf{z}$
10: **else**
11: $\beta = \rho/\rho 1$
12: $\mathbf{p} = \mathbf{z} + \beta\mathbf{p}$
13: **end if**
14: $\rho 1 = \rho$
15: $\mathbf{q} = \mathbf{A}\mathbf{p}$
16: $\alpha = \rho/(\mathbf{p} \cdot \mathbf{q})$
17: $\mathbf{x} = \mathbf{x} + \alpha\mathbf{p}$
18: $\mathbf{r} = \mathbf{r} - \alpha\mathbf{q}$
19: **until** convergence

the algorithm is the preconditioner solve step, which effectively receives the whole burden of the algorithm. In general, the condition number $\kappa_2(\mathbf{A})$ and the number of iterations grows as a function of the matrix dimension N. If $\mathbf{M}$ approximates $\mathbf{A}$ in some way, then $\mathbf{M}^{-1} \simeq \mathbf{A}^{-1}$ and $\kappa_2(\mathbf{M^{-1}A}) \simeq \kappa_2(\mathbf{I}) = 1$, which makes the PCG converge quickly as the number of iterations become independent of the matrix dimension (i.e. they are bounded by a constant, $\mathcal{O}(1)$). So the motivation behind preconditioning is to find a matrix $\mathbf{M}$ with the following properties: 1) the convergence rate of the preconditioned system $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$ is fast, and 2) a linear system involving $\mathbf{M}$ (i.e. $\mathbf{Mz} = \mathbf{r}$) can be solved much more efficiently (in $\mathcal{O}(N)$ or slightly higher number of operations) than the original system involving $\mathbf{A}$, in which case the performance of PCG will be optimal or very close to optimal.

An additional salient feature for a preconditioner matrix is the degree of parallelism that the solution of the linear system $\mathbf{Mz} = \mathbf{r}$ must entail. The last characteristic is quite significant in order allow for efficient mapping of the preconditioner solve step onto parallel architectures.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Preconditioning Algorithms

As was aforementioned, the preconditioner matrix $\mathbf{M}$ must provide a good approximation to the system matrix $\mathbf{A}$. The optimal preconditioner matrix is $\mathbf{M} = \mathbf{A}$, where the product $\mathbf{M}^{-1}\mathbf{A}$ has a condition number equal to 1. However, the preconditioner solve step requires solving the system $\mathbf{Mx} = \mathbf{b}$, which involves again the problem of the solution of a linear system with $\mathbf{A}$ as a system matrix.

As we can deduce, finding an optimal preconditioner is not trivial. A large number of research approaches focused on developing general purpose preconditioners. Although such preconditioners can be used as black boxes without specific knowledge of the underlying problem, their efficacy in terms of the convergence rate for every problem is not guaranteed. On the other hand, problem-tailored preconditioners can provide a great acceleration of the convergence rate. However, developing such a preconditioner requires that the underlying problem has a special structure, which is not the case for most problems.

The simplest preconditioner is a diagonal matrix whose diagonal entries are identical to those of $\mathbf{A}$. The preconditioner matrix is known as the diagonalor Jacobi preconditioner and its application requires inverting a diagonal matrix. Although inverting a diagonal matrix is a trivial process, the Jacobi preconditioner often offers mediocre results.

Another class of preconditioning algorithms are algorithms based on incomplete matrix factorizations. Incomplete LU preconditioning can be used in generic linear systems whereas Incomplete Cholesky preconditioning can be used in the case of an SPD system. Both are variants of the factorization methods (namely LU and Cholesky) that were described in Section 3.1. The Incomplete Cholesky algorithm creates a preconditioner matrix $\mathbf{M} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is a lower triangular matrix. Its main difference with the Cholesky factorization is that little or no fill-ins are allowed in the $\mathbf{L}$ factor. If no fill-ins are allowed, $\mathbf{L}$ is restricted to have the same sparsity pattern as $\mathbf{A}$ and all other elements are discarded. In this case, the solution of the preconditioner solve step $\mathbf{Mz} = \mathbf{r} \Rightarrow \mathbf{L}\mathbf{L}^T\mathbf{z} = \mathbf{r}$ is computed with forward

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

and backward substitution. Unfortunately, the Incomplete Cholesky is not always a stable preconditioner.

The need for more elaborate preconditioning algorithms enforced researchers to deviate from the standard practice of incomplete factorizations and use more advanced techniques like Sparse Approximate Inverse or Multigrid preconditioners. Sparse Approximate Inverse (SPAI) preconditioners [8] compute a preconditioner matrix $\mathbf{M} \approx \mathbf{A}^{-1}$ as the solution of the constrained minimization problem:

$$\min_{\mathbf{M} \in S} \| \mathbf{I} - \mathbf{AM} \|_F$$

where S is a set of sparsity pattern and $\|\|_F$ is the Frobenius norm of a matrix. The advantage of this method is that after construction, preconditioner's application (preconditioner solve step) requires only a matrix-vector multiplication as we already have $\mathbf{M}^{-1}$, making a Sparse Approximate Inverse preconditioner extremely efficient. However, the main problem with this type of preconditioners is that finding an appropriate sparsity pattern is not easy. An intuitive approach would be one to select the sparsity pattern of system matrix $\mathbf{A}$. However, this does not work well for complicated problems, as the preconditioner may require a larger number of non-zero elements in order to improve convergence.

On the other hand, multigrid algorithms [7] create a preconditioner by approximating the original matrix through a hierarchy of coarser matrices (or coarser grids). There are both geometric and algebraic approaches, with the latter taking advantage of the a-priori knowledge of the geometry of the underlying problem and the latter working solely on the system matrix. The main idea behind multigrid methods when used as preconditioners is to accelerate convergence of the iterative method by global correction from time to time. They are iterative methods on their own and consist of the following steps:

- Smoothing that aims at reducing high frequency errors, for example using a few iterations of the Gauss–Seidel method.
- Restriction, which is the downsampling of the residual error to a coarser grid.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

- Interpolation (or prolongation), which interpolates (maps) the correction computed on a coarser grid into a finer grid.

The main advantage of multigrid is that it often scales linearly with the number of discrete nodes used, which means that it can solve these problems to a given accuracy in a number of operations that is proportional to the number of unknowns. However, its main drawbacks is that multigrid is an iterative method on its own and that the prolongation and restriction operators are not rigorously defined for every problem. As a result, they can tax the ability for convergence of an iterative method when used as preconditioning mechanisms.

3.3 Solution of Linear Systems on Parallel Architectures

With the advent of massively parallel architectures, there has been an increasing demand for linear system solution algorithms that could take advantage of the computational resources that the former offer. Research works [18] and [49] present a parallel implementation of the LU and the Cholesky factorization direct algorithms on a GPU. Authors in [18] reduce the problem to a series of rasterization problems and use appropriate data representations to match the blocked rasterization order and cache pre-fetch technology of a GPU. They exploit high spatial coherence between elementary row operations and use fast parallel data transfer techniques to move data on GPUs. In the same context, authors in [49] organize blocks of nodes of a sparse matrix in the supernode data structure for GPU and propose a queue-based approach for the generation and scheduling of GPU tasks with dense linear algebraic operations in order to accelerate factorization. The main problems with the aforementioned approaches is that they entail a small degree of parallelism both in the factorization and the solution phase. As a result, they are not good candidates for mapping onto a parallel architecture. In addition, their large memory demands can exceed the main memory available on a GPU, making their execution infeasible.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

On the other hand, mapping an iterative method on a parallel architecture is a trivial process as both the matrix-vector and the vector-vector products entail a large degree of parallelism. The only delicate part is the preconditioner solve step, which is not amenable for mapping onto a parallel architecture if the solution of the preconditioner matrix $\mathbf{M}$ does not contain a large degree of parallelism. Incomplete factorization-based preconditioners require the construction of the preconditioner and the triangular solution steps. For both the Incomplete LU and the Incomplete Cholesky preconditioners, the factorization phase is amenable to porting on a parallel architecture. However, the triangular solution step exhibits a little degree of parallelism. As a result, in a parallel implementation that involves a GPU, the triangular solution step is performed at the CPU and the solution vector is transferred back to the GPU, which can greatly limit acceleration [23]. This is not the case for multigrid preconditioners that exhibit a large degree of parallelism, as the internal grid can be mapped on the computational grid of a GPU quite well. However, the drawbacks that were mentioned in Section 3.2.2 hinder its wide adoption as a parallel preconditioning method.

Based on the aforementioned observations, we can deduce that finding a preconditioner that will be able to accelerate convergence rate, while at the same time exhibit a large degree of parallelism is not trivial. The following chapters will describe three novel, extremely efficient and highly-parallel preconditioning approaches that in combination with an iterative method can enable analysis of very large-scale power delivery networks on massively parallel architectures.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Chapter 4

Power Grid Electrical Simulation

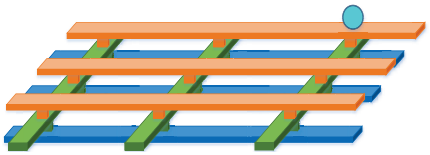
4.1 Power Grid Modeling

Power grid electrical simulation refers to the process of finding the response of the power delivery network when constant (DC analysis) or time-varying (transient analysis) sources are present. A power delivery network is a multi-layer network that comprises a number of metal lines, with alternating routing in horizontally and vertically directions. The lower-most metal layer connects the power delivery network with the logic gates, whereas the upper metal layer connects the network with the supply voltage. Each IC comprises two power delivery networks: one for the supply voltage (VDD network) and the other for the ground (GND network).

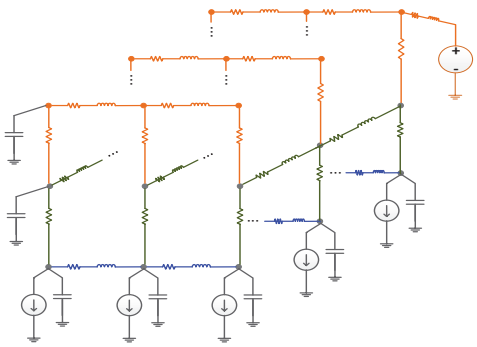
The typical model of a power grid is obtained by modeling each wire segment (between two contacts) as a resistance in series with an inductance, with capacitances to ground at both contact nodes. Each power bump is modeled as a voltage source while each logic gate is modeled as a current source. Fig. 4-1(i) depicts the geometry structure of a 3D power grid for the VDD supply with 3 layers and Fig. 4-1(ii) depicts its equivalent model for electrical analysis.

Let the electrical model of the power grid be composed of b composite R-L branches and N non-supply nodes. If we apply the Modified Nodal Analysis (MNA) method, we formulate the systems of linear equations (4.1) and (4.2), where the first is for DC and the second is for transient analysis:

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7



(i) Geometry structure of a 3D power grid with 3 layers. Vias represent connections between adjacent metal layers while the blue circle represents the solder bump.



(ii) Equivalent model for electrical analysis.

Figure 4-1: Example of a power delivery network with 3 horizontal and 3 vertical rails, along with its equivalent model for electrical analysis. The figure depicts only the VDD rails.

$$\mathbf{G}\mathbf{x} = \mathbf{e} \tag{4.1}$$

$$\tilde{\mathbf{G}}\mathbf{x}(t) + \tilde{\mathbf{C}}\dot{\mathbf{x}}(t) = \mathbf{e}(t) \tag{4.2}$$

where

$$\tilde{\mathbf{G}} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{rl} \\ -\mathbf{A}_{rl}^T & \mathbf{R}_b \end{bmatrix}$$

$$\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_b \end{bmatrix}$$

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{v}_n(t) \\ \mathbf{i}_b(t) \end{bmatrix}$$

$$\mathbf{e}(t) = \begin{bmatrix} \mathbf{e}_n(t) \\ \mathbf{0} \end{bmatrix}$$

In the systems described in (4.1) and (4.2), $\mathbf{A}_{rl}$ is the $N \times b$ incidence matrix of the

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

directed composite R-L branches (with elements $a_{ij} = \pm 1$ or $a_{ij} = 0$ depending on whether branch j leaves/enters or is not incident with node i), $\mathbf{v}_n(t)$, $\mathbf{i}_b(t)$ are the $N \times 1$ and $b \times 1$ vectors of node voltages and branch currents respectively, $\mathbf{e}_n(t)$ is a $N \times 1$ vector of excitations from independent sources at the nodes, $\mathbf{C}_n$ is a $N \times N$ diagonal matrix of the node capacitances, and $\mathbf{R}_b$, $\mathbf{L}_b$ are diagonal $b \times b$ matrices of the resistances and self-inductances of the composite R-L branches.

The system from (4.2) represents a non-linear system of equations. Using the Backward-Euler approximation we obtain from (4.2) the following discretized system of linear algebraic equations:

$$(\tilde{\mathbf{G}} + \frac{\tilde{\mathbf{C}}}{h_k})\mathbf{x}(h_k) = \mathbf{e}(h_k) + \frac{\tilde{\mathbf{C}}}{h_k}\mathbf{x}(h_{k-1}) \tag{4.3}$$

where $h_k,\ k = 1, \ldots$ is the chosen time-step that may in general vary during the analysis. By block-matrix operations on the above system we obtain the following system of coupled recursive equations [11]:

$$\begin{aligned}
&(\mathbf{A}_{rl}(\mathbf{R}_b + \frac{\mathbf{L}_b}{h_k})^{-1}\mathbf{A}_{rl}^T + \frac{\mathbf{C}_n}{h_k})\mathbf{v}_n(h_k) = \\
&\frac{\mathbf{C}_n}{h_k}\mathbf{v}_n(h_{k-1}) - (\mathbf{A}_{rl}(\mathbf{R}_b + \frac{\mathbf{L}_b}{h_k})^{-1}\frac{\mathbf{L}_b}{h_k}\mathbf{i}_b(h_{k-1}) + \mathbf{e}(h_k) && \text{(4.4i)} \\
&\mathbf{i}_b(h_k) = (\mathbf{R}_b + \frac{\mathbf{L}_b}{h_k})^{-1}\frac{\mathbf{L}_b}{h_k}\mathbf{i}_b(h_{k-1}) + (\mathbf{R}_b + \frac{\mathbf{L}_b}{h_k})^{-1}\mathbf{A}_{rl}^T\mathbf{v}_n(h_k) && \text{(4.4ii)}
\end{aligned}$$

At each time-step h_k we have to solve the $N \times N$ linear system (4.4i) with system matrix $\mathbf{A} \equiv \mathbf{A}_{rl}(\mathbf{R}_b + \frac{\mathbf{L}_b}{h_k})^{-1}\mathbf{A}_{rl}^T + \frac{\mathbf{C}_n}{h_k}$ and then find branch currents from (4.4ii). If we neglect inductances and model the grid as an RC circuit, the system (4.4) reduces to the following system:

$$(\mathbf{A}_{rl}\mathbf{R}_b^{-1}\mathbf{A}_{rl}^T + \frac{\mathbf{C}_n}{h_k})\mathbf{v}_n(h_k) = \frac{\mathbf{C}_n}{h_k}\mathbf{v}_n(h_{k-1}) + \mathbf{e}(h_k) \tag{4.5}$$

In both DC and transient analysis, the system matrix can be shown to be an SPD matrix, which means that efficient direct or iterative methods such as the

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Cholesky factorization or the method of PCG can be employed for its solution. However, the sheer size of power delivery networks from large-scale ICs mandates utilization of preconditioned iterative methods instead of direct methods due to the large computational and memory demands of the latter.

In addition, electrical simulation with variable time-step completely rules out direct methods as they require a matrix re-factorization each time the time-step is modified, thus increasing their computational demands especially for large-scale power delivery networks. In addition, a fixed time-step is almost never used in practice because it becomes very inefficient to constantly simulate during long intervals of low activity. All practical implementations of integration techniques for ODEs employ a variable or adaptive time-step mechanism [9]. In those cases, the reusability of matrix factorization in direct methods ceases to exist. This is true even for general-purpose preconditioning mechanisms that are based on matrix factorization (such as the Incomplete Cholesky) preconditioner that was described in Chapter 3. Although the computational demands of calculating the incomplete factorization are significantly lower than that for full factorization, this represents an additional overhead for the corresponding simulation algorithm.

As a result, finding a preconditioning mechanism that will be efficient for analysis with variable time-step both in terms of computational and memory demands is essential in order to allow for the wide adoption of iterative linear system solution algorithms. In addition, any preconditioning mechanism has to offer a significant degree of parallelism in order to allow for mapping onto parallel architectures.

Fortunately, as we will describe, matrices arising from power delivery networks can be well-approximated by preconditioners with special structure such that the number of iterations becomes bounded (or very slowly rising), while the preconditioned linear system $\mathbf{Mz} = \mathbf{r}$ can be solved by applying a Fast Transform in a near-optimal number of operations in a sequential implementation, and even less operations in a parallel environment (owing to the large parallel potential of Fast Transforms as well as other parallelization opportunities).

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

4.2 Related Work

The growing need to simulate large power grids resulted to the design and development of a large number of power grid analysis methods. These methods can be categorized into methods that exploit special structures (such as specific patterns) of the power delivery network and methods that utilize simulation in order to analyze the behavior of the power delivery network. The former category includes algorithms that obtain approximated solutions for the power delivery network, e.g. algorithms that apply the idea of multigrid techniques for solving partial differential equations for power grid analysis [20] [48] [26], Fast Poisson solvers [32], methods based on analysis in the frequency domain [19], stochastic-based algorithms [28], or model order reduction techniques [37] [22]. Although these methods allow for fast analysis of a power delivery network, they lack the desired accuracy (which is very important for late design stage analysis), they do not scale well with the size of the power grid, and in their majority do not provide a large degree of parallelism. As a result, the latter category of algorithms has attracted a lot of interest due to the ability for accurate characterization of the static and dynamic behavior of the power delivery network. Methods that fall into the aforementioned category include methods that utilize direct and iterative solution algorithms.

Direct solution methods are proposed in [45] and [34] that utilize either the LU or the Cholesky decomposition. Although they are robust and achieve the most accurate results, they suffer from a super-linear increase in computational and memory demands with the problem size. In addition, they provide limited room for parallelism, and as a result, they cannot utilize the computational resources of massively parallel architectures.

The need for simulation methods with small memory footprint and efficient parallel execution has led many researchers to deviate from the standard practice of direct factorization methods and present more suitable iterative methods. Research works [11] and [33] have proposed iterative solvers for efficient simulation of power delivery networks. Power grid analysis was first formulated as a symmetric positive

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

definite system to be solved by PCG in [11], but the preconditioner used was the general-purpose (and inefficient for specialized applications) incomplete Cholesky. A different pattern-based preconditioner was proposed in [33], but it is very simple and heuristic and does not appear to reduce the number of iterations significantly.

Recently, parallel architectures have been utilized to accelerate power grid analysis. Authors in [34] have proposed domain decomposition as a parallel technique for analysis on multi-core architectures. GPUs are used in research approaches [32], [16], and [17] as parallel platforms that enable tackling power analysis of large-scale power networks. Authors in [17] propose multi-grid as a solution method for power grid analysis and they use multi-core and massively parallel Single Instruction Multiple Thread (SIMT) platforms to tackle power grid analysis, while authors in [32] formulate the traditional linear system as a special two-dimension Poisson equation and solve it using analytical expressions based on the FFT algorithm, with GPUs being used to further speed up the algorithm. However, both [32] and [17] only solve very regular grid structures with specialized techniques, which can limit their effectiveness for irregular power delivery networks that are found in late design stages. Instead, we propose to use such a regular structure as preconditioner in order to solve any practical (and possibly irregular) power delivery network.

Preconditioning has lately drawn attention as a method for efficiently tackling the analysis of large-scale and irregular power grid designs. Such a possibility is the topic of research works [16], [42], [25], and [41] for power grid analysis, and [29] in the context of IC thermal simulation. In [16] and [42], the preconditioning has been carried out by multigrid techniques. However, when used as preconditioner for iterative methods, multigrid is not very efficient because it is an iterative method by itself, and also solves a system approximately which can hinder the convergence of PCG. Moreover, some operations of multigrid are not always well-defined (e.g. mapping by interpolation from coarser to finer grids and back, and correction of solutions), and the construction of approximate matrices for all coarser grids is an expensive setup phase which has to be repeated every time the system is reconstructed in each time-step change during transient analysis. Our approaches

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

for preconditioning, based on the application of a Fast Transform, involves a much more straightforward and inexpensive implementation and reconstruction phase for transient simulation, while it also provides analytical solution of the preconditioned system (since it is actually a fast direct method). On the other hand, research work [25] formulates the problem of power grid analysis as a non-SPD problem and utilizes the GMRES iterative solution algorithm with an Incomplete LU preconditioner. The drawbacks of the method is the increased complexity as the GMRES algorithm exhibits a higher computational complexity than the PCG algorithm and the limited parallelism in the preconditioner solve step, due to the solution of the triangular factors that form the preconditioner, which render our proposed algorithms more efficient.

The approach in [41] is the one closest to ours, in the sense that it uses a Fast Poisson-based preconditioner to accelerate the convergence rate of CG. However, the proposed technique is based on the presumed existence of special areas in the grid with zero voltage drop as boundary condition, in order to formulate so-called "Poisson blocks" with Toeplitz matrix structure, while our approach does not necessitate such an assumption.

In the same context, authors in [46] and [38] present a support graph-based and a Random Walk-based preconditioner respectively that can provide a significant acceleration to the convergence rate of an iterative method. However, applying these preconditioners requires the solution of a triangular system which can hinder preconditioner's applicability on parallel architectures due to the limited parallelism of triangular solution algorithms (where the obtained speedup cannot be larger than $2X$, as is reported in [40]. On the contrary, applying our Fast Transform preconditioners has greater potential for parallelism than both multigrid and triangular solution algorithms, especially on GPUs [21] [3].

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

4.3 Fast Transform Solvers for Networks with Special Structure

Fast Transform solvers originate from the solution of the Poisson equation and are a special category of direct methods for the solution of linear systems of the form $\mathbf{Mz} = \mathbf{r}$, where the system matrix $\mathbf{M}$ has a special structure. Their main characteristics are the optimal (near-linear) complexity and the large degree of multi-level parallelism. This section describes two Fast Transform solver algorithms for 2D and 3D networks with special structure.

4.3.1 Fast Transform Solvers for 2D Networks

Let $\mathbf{M}$ be a $N \times N$ block-tridiagonal matrix with m blocks of size $n \times n$ each (overall $N = mn$), which has the following form:

$$\mathbf{M} = \begin{bmatrix} \mathbf{T}_1 & \gamma_1\mathbf{I} & & & \\ \gamma_1\mathbf{I} & \mathbf{T}_2 & \gamma_2\mathbf{I} & & \\ & \cdot & \cdot & \cdot & \\ & & \gamma_{m-2}\mathbf{I} & \mathbf{T}_{m-1} & \gamma_{m-1}\mathbf{I} \\ & & & \gamma_{m-1}\mathbf{I} & \mathbf{T}_m \end{bmatrix} \tag{4.6}$$

where $\mathbf{I}$ is the $n \times n$ identity matrix and $\mathbf{T}_i$, $i = 1, \ldots, m$, are $n \times n$ tridiagonal matrices of the form:

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$$\mathbf{T}_i = \begin{bmatrix} \alpha_i+\beta_i & -\alpha_i & & & & \\ -\alpha_i & 2\alpha_i+\beta_i & -\alpha_i & & & \\ & \cdot & \cdot & \cdot & & \\ & & -\alpha_i & 2\alpha_i+\beta_i & -\alpha_i & \\ & & & -\alpha_i & \alpha_i+\beta_i & \end{bmatrix}$$
$$= \alpha_i \begin{bmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & \cdot & \cdot & \cdot & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 1 & \end{bmatrix} + \beta_i \mathbf{I} \tag{4.7}$$

We will describe an algorithm for the solution of a linear system $\mathbf{Mz} = \mathbf{r}$ with matrix $\mathbf{M}$ of the form (4.6), by the use of a Fast Transform solver in $\mathcal{O}(mn \log n) = \mathcal{O}(N \log n)$ operations. Such a solution is based on the fact that the eigendecomposition of the tridiagonal matrices $\mathbf{T}_i$ is known beforehand, and that the matrices of eigenvectors that diagonalize $\mathbf{T}_i$ are matrices that correspond to a Fast Transform. More specifically, it can be shown (see Appendix A) that each $\mathbf{T}_i$ has n distinct eigenvalues $\lambda_{i,j}$, $j = 1, \ldots, n$, which are given by:

$$\lambda_{i,j} = \beta_i + 4\alpha_i \sin^2 \frac{(j-1)\pi}{2n} = \beta_i + \alpha_i (2 - 2\cos \frac{(j-1)\pi}{n}) \tag{4.8}$$

as well as a set of n orthonormal eigenvectors $\mathbf{q}_j$, $j = 1, \ldots, n$, with elements:

$$q_{j,k} = \begin{cases} \sqrt{\frac{1}{n}} \cos \frac{(2k-1)(j-1)\pi}{2n} & j = 1, \quad k = 1, \ldots, n \\ \sqrt{\frac{2}{n}} \cos \frac{(2k-1)(j-1)\pi}{2n} & j = 2, \ldots, n, \quad k = 1, \ldots, n \end{cases} \tag{4.9}$$

Note that the $\mathbf{q}_j$ do not depend on the values of α_i and β_i, and are the same for every matrix $\mathbf{T}_i$. If $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ denotes the matrix whose columns are the eigenvectors $\mathbf{q}_j$, then due to the eigen-decomposition of $\mathbf{T}_i$ we have $\mathbf{Q}^T \mathbf{T}_i \mathbf{Q} = \mathbf{\Lambda}_i = diag(\lambda_{i,1}, \ldots, \lambda_{i,n})$. By exploiting this diagonalization of the matrices $\mathbf{T}_i$, the system

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$\mathbf{Mz} = \mathbf{r}$ with $\mathbf{M}$ of the form (4.6) is equivalent to the following system (due to $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$):

$$\begin{bmatrix} \mathbf{Q}^T & & \\ & \ddots & \\ & & \mathbf{Q}^T \end{bmatrix} \mathbf{M} \begin{bmatrix} \mathbf{Q} & & \\ & \ddots & \\ & & \mathbf{Q} \end{bmatrix} \begin{bmatrix} \mathbf{Q}^T & & \\ & \ddots & \\ & & \mathbf{Q}^T \end{bmatrix} \mathbf{z}$$
$$= \begin{bmatrix} \mathbf{Q}^T & & \\ & \ddots & \\ & & \mathbf{Q}^T \end{bmatrix} \mathbf{r} \Leftrightarrow$$

$$\begin{bmatrix} \mathbf{\Lambda}_1 & \gamma_1\mathbf{I} & & & & \\ \gamma_1\mathbf{I} & \mathbf{\Lambda}_2 & \gamma_2\mathbf{I} & & & \\ & . & . & . & & \\ & & \gamma_{m-2}\mathbf{I} & \mathbf{\Lambda}_{m-1} & \gamma_{m-1}\mathbf{I} & \\ & & & \gamma_{m-1}\mathbf{I} & \mathbf{\Lambda}_m & \end{bmatrix} \tilde{\mathbf{z}} = \tilde{\mathbf{r}} \tag{4.10}$$

where

$$\tilde{\mathbf{z}} = \begin{bmatrix} \mathbf{Q}^T & & \\ & \ddots & \\ & & \mathbf{Q}^T \end{bmatrix} \mathbf{z}, \quad \tilde{\mathbf{r}} = \begin{bmatrix} \mathbf{Q}^T & & \\ & \ddots & \\ & & \mathbf{Q}^T \end{bmatrix} \mathbf{r}$$

If the $N \times 1$ vectors $\mathbf{r}$, $\mathbf{z}$, $\tilde{\mathbf{r}}$, $\tilde{\mathbf{z}}$ are also partitioned into m blocks of size $n \times 1$ each, i.e.

$$\mathbf{r} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_m \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_m \end{bmatrix}, \quad \tilde{\mathbf{r}} = \begin{bmatrix} \tilde{\mathbf{r}}_1 \\ \vdots \\ \tilde{\mathbf{r}}_m \end{bmatrix}, \quad \tilde{\mathbf{z}} = \begin{bmatrix} \tilde{\mathbf{z}}_1 \\ \vdots \\ \tilde{\mathbf{z}}_m \end{bmatrix}$$

then we have: $\tilde{\mathbf{r}}_i = \mathbf{Q}^T\mathbf{r}_i$ and $\tilde{\mathbf{z}}_i = \mathbf{Q}^T\mathbf{z}_i \Leftrightarrow \mathbf{z}_i = \mathbf{Q}\tilde{\mathbf{z}}_i,\ i = 1, \ldots, m$.

However, it can be shown [36] that each product $\mathbf{Q}^T\mathbf{r}_i = \tilde{\mathbf{r}}_i$ corresponds to a Discrete Cosine Transform (DCT) of type-II (DCT-II) on $\mathbf{r}_i$, and each product

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$\mathbf{Q}\tilde{\mathbf{z}}_i = \mathbf{z}_i$ corresponds to an Inverse Discrete Cosine Transform (IDCT) of type-II (IDCT-II) on $\tilde{\mathbf{z}}_i$. This means that the computation of the whole vector $\tilde{\mathbf{r}}$ from $\mathbf{r}$ amounts to m independent DCT-II transforms of size n, and the computation of the whole vector $\mathbf{z}$ from $\tilde{\mathbf{z}}$ amounts to m independent IDCT-II transforms of size n. A modification of FFT can be employed for each of the m independent DCT-II/IDCT-II transforms [36], giving a total operation count of $\mathcal{O}(mn\log n) = \mathcal{O}(N\log n)$.

If now $\mathbf{P}$ is a permutation matrix that reorders the elements of a vector or the rows of a matrix as $1, n+1, 2n+1, \ldots, (m-1)n+1,\ 2, n+2, 2n+2, \ldots, (m-1)n+2,\ \ldots,\ n, n+n, 2n+n, \ldots, (m-1)n+n$, and $\mathbf{P}^T$ is the inverse permutation matrix, then the system (4.10) is further equivalent to:

$$\mathbf{P}\begin{bmatrix} \mathbf{\Lambda}_1 & \gamma_1\mathbf{I} & & & \\ \gamma_1\mathbf{I} & \mathbf{\Lambda}_2 & \gamma_2\mathbf{I} & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{m-2}\mathbf{I} & \mathbf{\Lambda}_{m-1} & \gamma_{m-1}\mathbf{I} \\ & & & \gamma_{m-1}\mathbf{I} & \mathbf{\Lambda}_m \end{bmatrix}\mathbf{P}^{\mathbf{T}}\mathbf{P}\tilde{\mathbf{z}} = \mathbf{P}\tilde{\mathbf{r}} \Leftrightarrow$$

$$\begin{bmatrix} \tilde{\mathbf{T}}_1 & & & \\ & \tilde{\mathbf{T}}_2 & & \\ & & \ddots & \\ & & & \tilde{\mathbf{T}}_n \end{bmatrix}\tilde{\mathbf{z}}^P = \tilde{\mathbf{r}}^P \tag{4.11}$$

where

$$\tilde{\mathbf{T}}_j = \begin{bmatrix} \lambda_{1,j} & \gamma_1 & & & \\ \gamma_1 & \lambda_{2,j} & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{m-2} & \lambda_{m-1,j} & \gamma_{m-1} \\ & & & \gamma_{m-1} & \lambda_{m,j} \end{bmatrix} \tag{4.12}$$

and $\tilde{\mathbf{z}}^P = \mathbf{P}\tilde{\mathbf{z}}$, $\tilde{\mathbf{r}}^P = \mathbf{P}\tilde{\mathbf{r}}$. If the $N \times 1$ vectors $\tilde{\mathbf{z}}^P$, $\tilde{\mathbf{r}}^P$ are partitioned into n blocks

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$\tilde{\mathbf{z}}_j^P, \tilde{\mathbf{r}}_j^P$ of size $m \times 1$ each, then the system (4.11) effectively represents n independent tridiagonal systems $\mathbf{T}_j \tilde{\mathbf{z}}_j^P = \tilde{\mathbf{r}}_j^P$ of size m which can be solved with respect to the blocks $\tilde{\mathbf{z}}_j^P,\ j = 1, \ldots, n$ (to produce the whole vector $\tilde{\mathbf{z}}^P$) in a total of $\mathcal{O}(mn) = \mathcal{O}(N)$ operations. For each such system the coefficient matrix (4.12) is known beforehand and is determined exclusively by the eigenvalues (4.8) and the values γ_i of matrix $\mathbf{M}$, while the right-hand side (RHS) vector $\tilde{\mathbf{r}}_j^P$ is composed of specific components of (DCT-II)-transformed blocks of vector $\mathbf{r}$. The equivalence of the system $\mathbf{Mz} = \mathbf{r}$, with $\mathbf{M}$ as in (4.6), to the system (4.11) gives a procedure for fast solution of $\mathbf{Mz} = \mathbf{r}$ which is described in Algorithm 6.

4.3.2 Fast Transform Solvers for 3D Networks

The methodology described in Section 4.3.1 can be extended and applied for the solution of 3D networks. Let $\mathbf{M}$ be a $N \times N$ block-tridiagonal matrix with l blocks of size $mn \times mn$ each (overall $N = lmn$), where l is very small (typically between 2 and 8), since for power grid matrices it corresponds to the number of metal layers. For ease of presentation we will assume $l = 4$ in the following, but the development is perfectly generalizable to an arbitrary number l. In particular, matrix $\mathbf{M}$ has the following form:

Algorithm 6 Fast Transform algorithm for the preconditioner solve step $\mathbf{Mz} = \mathbf{r}$, where $\mathbf{M}$ is of the form (4.6)

1: Partition the RHS vector $\mathbf{r}$ into m blocks $\mathbf{r}_i$ of size n, and perform DCT-II transform ($\mathbf{Q}^T \mathbf{r}_i$) on each block to obtain transformed vector $\tilde{\mathbf{r}}$
2: Permute vector $\tilde{\mathbf{r}}$ by permutation $\mathbf{P}$, which orders elements as $1, n+1, \ldots, (m-1)n+1,\ 2, n+2, \ldots, (m-1)n+2,\ \ldots,\ n, n+n, \ldots, (m-1)n+n$, in order to obtain vector $\tilde{\mathbf{r}}^P$
3: Solve the n tridiagonal systems (4.11) with known coefficient matrices (4.12), in order to obtain vector $\tilde{\mathbf{z}}^P$.
4: Apply inverse permutation $\mathbf{P}^T$ on vector $\tilde{\mathbf{z}}^P$ so as to obtain vector $\tilde{\mathbf{z}}$.
5: Partition vector $\tilde{\mathbf{z}}$ into m blocks $\tilde{\mathbf{z}}_i$ of size n, and perform IDCT-II transform ($\mathbf{Q}\tilde{\mathbf{z}}_i$) on each block to obtain final solution vector $\mathbf{z}$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & \gamma_1\mathbf{I}_{mn} & & \\ \gamma_1\mathbf{I}_{mn} & \mathbf{M}_2 & \gamma_2\mathbf{I}_{mn} & \\ & \gamma_2\mathbf{I}_{mn} & \mathbf{M}_3 & \gamma_3\mathbf{I}_{mn} \\ & & \gamma_3\mathbf{I}_{mn} & \mathbf{M}_4 \end{bmatrix} \tag{4.13}$$

where $\mathbf{I}_{mn}$ is the $mn \times mn$ identity matrix, and $\mathbf{M}_i$, $i = 1, 2, 3, 4$, are alternating block-diagonal and block-tridiagonal $mn \times mn$ matrices with m blocks of size $n \times n$ which have the forms:

$$\mathbf{M}_i = diag(\mathbf{T}_i, \ldots, \mathbf{T}_i), \quad i = 1, 3 \tag{4.14}$$

$$\mathbf{M}_i = \begin{bmatrix} (\alpha_i+\beta_i)\mathbf{I}_n & -\alpha_i\mathbf{I}_n & & & \\ -\alpha_i\mathbf{I}_n & (2\alpha_i+\beta_i)\mathbf{I}_n & -\alpha_i\mathbf{I}_n & & \\ & . & . & . & \\ & & & (2\alpha_i+\beta_i)\mathbf{I}_n & -\alpha_i\mathbf{I}_n \\ & & & -\alpha_i\mathbf{I}_n & (\alpha_i+\beta_i)\mathbf{I}_n \end{bmatrix}, \; i = 2, 4 \tag{4.15}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix, and $\mathbf{T}_i$, $i = 1, 3$ have the form (4.7). Thus, the eigenvalues and eigenvectors of the diagonal blocks of $\mathbf{M}_i$, $i = 1, 3$ are the same as those of $\mathbf{T}_i$, and are given by (4.8) and (4.9) respectively. By a similar reasoning as in (4.10), the linear system $\mathbf{Mz} = \mathbf{r}$ with $\mathbf{M}$ of the form (4.13) is equivalent to the following:

$$\begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{M} \begin{bmatrix} \mathbf{Q}_n & & \\ & \ddots & \\ & & \mathbf{Q}_n \end{bmatrix} \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{z}$$
$$= \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{r} \Leftrightarrow$$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$$\begin{bmatrix} \tilde{\mathbf{M}}_1 & \gamma_1\mathbf{I}_{mn} & & \\ \gamma_1\mathbf{I}_{mn} & \mathbf{M}_2 & \gamma_2\mathbf{I}_{mn} & \\ & \gamma_2\mathbf{I}_{mn} & \tilde{\mathbf{M}}_3 & \gamma_3\mathbf{I}_{mn} \\ & & \gamma_3\mathbf{I}_{mn} & \mathbf{M}_4 \end{bmatrix} \tilde{\mathbf{z}} = \tilde{\mathbf{r}} \tag{4.16}$$

where $\tilde{\mathbf{M}}_i = diag(\mathbf{\Lambda}_i, \ldots, \mathbf{\Lambda}_i),\ i = 1, 3$, and

$$\tilde{\mathbf{z}} = \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{z}, \quad \tilde{\mathbf{r}} = \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{r}$$

If the $N \times 1$ vectors $\mathbf{r}$, $\mathbf{z}$, $\tilde{\mathbf{r}}$, $\tilde{\mathbf{z}}$ are themselves partitioned into lm sub-vectors (blocks) of size $n \times 1$ each, i.e.

$$\mathbf{r} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_{lm} \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_{lm} \end{bmatrix}, \quad \tilde{\mathbf{r}} = \begin{bmatrix} \tilde{\mathbf{r}}_1 \\ \vdots \\ \tilde{\mathbf{r}}_{lm} \end{bmatrix}, \quad \tilde{\mathbf{z}} = \begin{bmatrix} \tilde{\mathbf{z}}_1 \\ \vdots \\ \tilde{\mathbf{z}}_{lm} \end{bmatrix}$$

then we have $\tilde{\mathbf{r}}_i = \mathbf{Q}_n^T\mathbf{r}_i$, and $\tilde{\mathbf{z}}_i = \mathbf{Q}_n^T\mathbf{z}_i \Leftrightarrow \mathbf{z}_i = \mathbf{Q}_n\tilde{\mathbf{z}}_i,\ i = 1, \ldots, lm$.

If $\mathbf{P}$ is again the permutation matrix of size $mn \times mn$ that reorders the elements of a vector or the rows of a matrix as $1, n+1, \ldots, (m-1)n+1,\ 2, n+2, \ldots, (m-1)n+2,\ \ldots,\ n, n+n, \ldots, (m-1)n+n$, and $\mathbf{P}_1$, $\mathbf{P}_1^T$ denote the block-diagonal $N \times N$ permutation matrices $\mathbf{P}_1 = diag(\mathbf{P}, \mathbf{P}, \mathbf{P}, \mathbf{P})$ and $\mathbf{P}_1^T = diag(\mathbf{P}^T, \mathbf{P}^T, \mathbf{P}^T, \mathbf{P}^T)$ then the system (4.16) is further equivalent to:

$$\mathbf{P}_1 \begin{bmatrix} \tilde{\mathbf{M}}_1 & \gamma_1\mathbf{I}_{mn} & & \\ \gamma_1\mathbf{I}_{mn} & \mathbf{M}_2 & \gamma_2\mathbf{I}_{mn} & \\ & \gamma_2\mathbf{I}_{mn} & \tilde{\mathbf{M}}_3 & \gamma_3\mathbf{I}_{mn} \\ & & \gamma_3\mathbf{I}_{mn} & \mathbf{M}_4 \end{bmatrix} \mathbf{P}_1^T\mathbf{P}_1\tilde{\mathbf{z}} = \mathbf{P}_1\tilde{\mathbf{r}} \Leftrightarrow$$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$$\begin{bmatrix} \mathbf{D}_1 & \gamma_1\mathbf{I}_{mn} & & \\ \gamma_1\mathbf{I}_{mn} & \mathbf{D}_2 & \gamma_2\mathbf{I}_{mn} & \\ & \gamma_2\mathbf{I}_{mn} & \mathbf{D}_3 & \gamma_3\mathbf{I}_{mn} \\ & & \gamma_3\mathbf{I}_{mn} & \mathbf{D}_4 \end{bmatrix} \tilde{\mathbf{z}}^{\mathbf{P}_1} = \tilde{\mathbf{r}}^{\mathbf{P}_1} \tag{4.17}$$

where

$$\mathbf{D}_i = diag(\lambda_{i,1}\mathbf{I}_m, \ldots, \lambda_{i,n}\mathbf{I}_m), \quad i = 1, 3$$
$$\mathbf{D}_i = diag(\mathbf{T}_i, \ldots, \mathbf{T}_i), \quad i = 2, 4$$

and $\tilde{\mathbf{z}}^{\mathbf{P}_1} = \mathbf{P}_1\tilde{\mathbf{z}}$, $\tilde{\mathbf{r}}^{\mathbf{P}_1} = \mathbf{P}_1\tilde{\mathbf{r}}$. In the above, $\mathbf{T}_i$, $i = 2, 4$, are $m \times m$ tridiagonal matrices of the form (4.7), each having m distinct eigenvalues as in (4.8) (with m in place of n), and m orthonormal eigenvectors as in (4.9) (with m in place of n). If $\mathbf{Q}_m = [\mathbf{q}_1, \ldots, \mathbf{q}_m]$ is the common matrix of eigenvectors for $\mathbf{T}_i$, $i = 2, 4$, and $\mathbf{Q}_m^T\mathbf{T}_i\mathbf{Q}_m = \mathbf{\Lambda}_i = diag(\lambda_{i,1}, \ldots, \lambda_{i,m})$ is the corresponding diagonalization, then the system (4.17) is further equivalent to:

$$\begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \begin{bmatrix} \mathbf{D}_1 & \gamma_1\mathbf{I}_{mn} & & \\ \gamma_1\mathbf{I}_{mn} & \mathbf{D}_2 & \gamma_2\mathbf{I}_{mn} & \\ & \gamma_2\mathbf{I}_{mn} & \mathbf{D}_3 & \gamma_3\mathbf{I}_{mn} \\ & & \gamma_3\mathbf{I}_{mn} & \mathbf{D}_4 \end{bmatrix} \begin{bmatrix} \mathbf{Q}_m & & \\ & \ddots & \\ & & \mathbf{Q}_m \end{bmatrix} \cdot$$
$$\cdot \begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \tilde{\mathbf{z}}^{\mathbf{P}_1} = \begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \tilde{\mathbf{r}}^{\mathbf{P}_1} \Leftrightarrow$$
$$\begin{bmatrix} \mathbf{D}_1 & \gamma_1\mathbf{I}_{mn} & & \\ \gamma_1\mathbf{I}_{mn} & \tilde{\mathbf{D}}_2 & \gamma_2\mathbf{I}_{mn} & \\ & \gamma_2\mathbf{I}_{mn} & \mathbf{D}_3 & \gamma_3\mathbf{I}_{mn} \\ & & \gamma_3\mathbf{I}_{mn} & \tilde{\mathbf{D}}_4 \end{bmatrix} \tilde{\tilde{\mathbf{z}}} = \tilde{\tilde{\mathbf{r}}} \tag{4.18}$$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

where $\tilde{\mathbf{D}}_i = diag(\mathbf{\Lambda}_i, \ldots, \mathbf{\Lambda}_i),\ i = 2, 4$, and

$$\tilde{\tilde{\mathbf{z}}} = \begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \tilde{\mathbf{z}}^{\mathbf{P}_1}, \quad \tilde{\tilde{\mathbf{r}}} = \begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \tilde{\mathbf{r}}^{\mathbf{P}_1}$$

In a similar fashion as previously, the $N \times 1$ vectors $\tilde{\mathbf{r}}^{\mathbf{P}_1}$, $\tilde{\mathbf{z}}^{\mathbf{P}_1}$, $\tilde{\tilde{\mathbf{r}}}$, $\tilde{\tilde{\mathbf{z}}}$ can be partitioned into ln sub-vectors of size $m \times 1$ each, and then the computation of $\tilde{\tilde{\mathbf{r}}}$ from $\tilde{\mathbf{r}}^{\mathbf{P}_1}$ amounts to ln independent DCT-II transforms of size m, and the computation of $\tilde{\mathbf{z}}^{\mathbf{P}_1}$ from $\tilde{\tilde{\mathbf{z}}}$ amounts to ln independent IDCT-II transforms of size m, leading to a serial operation count of $\mathcal{O}(lnm \log m) = \mathcal{O}(N \log m)$.

If we now denote by $\mathbf{P}_2$ the $N \times N$ permutation matrix that reorders the elements of a vector or the rows of a matrix as $1, mn+1, \ldots, (l-1)mn+1,\ 2, mn+2, \ldots, (l-1)mn+2,\ \ldots,\ mn, mn+mn, \ldots, (l-1)mn+mn$, and by $\mathbf{P}_2^T$ the inverse permutation matrix, then the system (4.18) is eventually equivalent to:

$$\mathbf{P}_2 \begin{bmatrix} \mathbf{D}_1 & \gamma_1 \mathbf{I}_{mn} & & \\ \gamma_1 \mathbf{I}_{mn} & \tilde{\mathbf{D}}_2 & \gamma_2 \mathbf{I}_{mn} & \\ & \gamma_2 \mathbf{I}_{mn} & \mathbf{D}_3 & \gamma_3 \mathbf{I}_{mn} \\ & & \gamma_3 \mathbf{I}_{mn} & \tilde{\mathbf{D}}_4 \end{bmatrix} \mathbf{P}_2^T \mathbf{P}_2 \tilde{\tilde{\mathbf{z}}} = \mathbf{P}_2 \tilde{\tilde{\mathbf{r}}} \Leftrightarrow$$

$$diag(\tilde{\mathbf{T}}_{1,1}, \tilde{\mathbf{T}}_{1,2}, \ldots, \tilde{\mathbf{T}}_{1,m}, \tilde{\mathbf{T}}_{2,1}, \ldots, \tilde{\mathbf{T}}_{2,m}, \ldots, \tilde{\mathbf{T}}_{n,m}) \tilde{\tilde{\mathbf{z}}}^{P_2} = \tilde{\tilde{\mathbf{r}}}^{P_2} \tag{4.19}$$

where

$$\tilde{\mathbf{T}}_{i,j} = \begin{bmatrix} \lambda_{1,i} & \gamma_1 & & \\ \gamma_1 & \lambda_{2,j} & \gamma_2 & \\ & \gamma_2 & \lambda_{3,i} & \gamma_3 \\ & & \gamma_3 & \lambda_{4,j} \end{bmatrix}, \quad i = 1, \ldots, n,\ j = 1, \ldots, m \tag{4.20}$$

and $\tilde{\tilde{\mathbf{z}}}^{P_2} = \mathbf{P}_2 \tilde{\tilde{\mathbf{z}}}$, $\tilde{\tilde{\mathbf{r}}}^{P_2} = \mathbf{P}_2 \tilde{\tilde{\mathbf{r}}}$. If the $N \times 1$ vectors $\tilde{\tilde{\mathbf{z}}}^{P_2}$, $\tilde{\tilde{\mathbf{r}}}^{P_2}$ are partitioned into mn sub-vectors $\tilde{\tilde{\mathbf{z}}}^{P_2}_{(i-1)m+j}, \tilde{\tilde{\mathbf{r}}}^{P_2}_{(i-1)m+j}$ of size $l \times 1$ each ($i = 1, \ldots, n,\ j = 1, \ldots, m$), then the system (4.19) effectively represents mn independent tridiagonal systems

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Algorithm 7 Fast Transform algorithm for the preconditioner solve step $\mathbf{Mz} = \mathbf{r}$, where $\mathbf{M}$ is of the form (4.13)

1: Partition the RHS vector $\mathbf{r}$ into lm sub-vectors $\mathbf{r}_i$ of size n, and perform DCT-II transform ($\mathbf{Q}_n^T \mathbf{r}_i$) on each sub-vector to obtain transformed vector $\tilde{\mathbf{r}}$.
2: Partition vector $\tilde{\mathbf{r}}$ into l sub-vectors $\tilde{\mathbf{r}}_i$ of size mn, and permute each sub-vector by permutation $\mathbf{P}$, which orders elements as $1, n+1, \ldots, (m-1)n+1,\ 2, n+2, \ldots, (m-1)n+2,\ \ldots,\ n, n+n, \ldots, (m-1)n+n$, in order to obtain vector $\tilde{\mathbf{r}}^{\mathbf{P}_1}$.
3: Partition vector $\tilde{\mathbf{r}}^{\mathbf{P}_1}$ into ln sub-vectors $\tilde{\mathbf{r}}_i^{\mathbf{P}_1}$ of size m, and perform DCT-II transform ($\mathbf{Q}_m^T \tilde{\mathbf{r}}_i^{\mathbf{P}_1}$) on each sub-vector to obtain transformed vector $\tilde{\tilde{\mathbf{r}}}$.
4: Permute vector $\tilde{\tilde{\mathbf{r}}}$ by applying permutation $\mathbf{P}_2$, which orders elements as $1, mn+1, 2mn+1, \ldots, (l-1)mn+1,\ 2, mn+2, 2mn+2, \ldots, (l-1)mn+2,\ \ldots,\ mn, mn+mn, 2mn+mn, \ldots, (l-1)mn+mn$, in order to obtain vector $\tilde{\tilde{\mathbf{r}}}^{P_2}$.
5: Solve the mn tridiagonal systems (4.19) with known coefficient matrices (4.20), in order to obtain vector $\tilde{\tilde{\mathbf{z}}}^{P_2}$.
6: Apply inverse permutation $\mathbf{P}_2^T$ on vector $\tilde{\tilde{\mathbf{z}}}^{P_2}$ so as to obtain vector $\tilde{\tilde{\mathbf{z}}}$.
7: Partition vector $\tilde{\tilde{\mathbf{z}}}$ into ln sub-vectors $\tilde{\tilde{\mathbf{z}}}_i$ of size m, and perform IDCT-II transform ($\mathbf{Q}_m \tilde{\tilde{\mathbf{z}}}_i$) on each sub-vector to obtain vector $\tilde{\mathbf{z}}^{\mathbf{P}_1}$.
8: Partition vector $\tilde{\mathbf{z}}^{\mathbf{P}_1}$ into l sub-vectors $\tilde{\mathbf{z}}_i^{\mathbf{P}_1}$ of size mn, and apply inverse permutation $\mathbf{P}^T$ on each sub-vector to obtain vector $\tilde{\mathbf{z}}$.
9: Partition vector $\tilde{\mathbf{z}}$ into lm sub-vectors $\tilde{\mathbf{z}}_i$ of size n, and perform IDCT-II transform ($\mathbf{Q}_n \tilde{\mathbf{z}}_i$) on each sub-vector to obtain final solution vector $\mathbf{z}$.

$\tilde{\mathbf{T}}_{i,j} \tilde{\tilde{\mathbf{z}}}^{P_2}_{(i-1)m+j} = \tilde{\tilde{\mathbf{r}}}^{P_2}_{(i-1)m+j}$ of size l which can be solved with respect to the sub-vectors $\tilde{\tilde{\mathbf{z}}}^{P_2}_{(i-1)m+j}$ (to produce the whole vector $\tilde{\tilde{\mathbf{z}}}^{P_2}$) in a total of $\mathcal{O}(lmn) = \mathcal{O}(N)$ serial operations. For each such system, the coefficient matrix (4.20) is known beforehand and is determined exclusively by the eigenvalues (4.8) and the values γ_i of the matrix $\mathbf{M}$.

The equivalence of the system $\mathbf{Mz} = \mathbf{r}$, with $\mathbf{M}$ as in (4.13), to the system (4.19), gives a procedure for solution of $\mathbf{Mz} = \mathbf{r}$ in a near-optimal number of $\mathcal{O}(N) + \mathcal{O}(N(\log n + \log m)) = \mathcal{O}(N \log (nm))$ operations, which is described in Algorithm 7. Note that apart from the near-optimal serial complexity, both algorithms entail a great amount of task-level parallelism. The m DCT-II/IDCT-II transforms and the n tridiagonal systems of Algorithm 6, as well as the lm first-level DCT-II/IDCT-II transforms respectively, the ln second-level DCT-II/IDCT-II transforms, and the mn tridiagonal systems of Algorithm 7 are completely independent to each other and can be executed in parallel. Furthermore, FFT is a highly-parallel algorithm by

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

itself, allowing for further acceleration of the individual transforms when executed on parallel platforms. These issues are further examined in Section 4.4.2, in the context of our proposed approach for power grid analysis.

4.4 Proposed Methodology for Power Grid Analysis

4.4.1 Preconditioner Construction and Storage

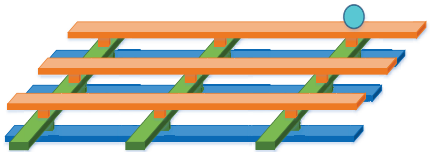
As mentioned in Section 3.2.2, the intuition behind preconditioner's formulation is to create a matrix $\mathbf{M}$ that will approximate the system matrix $\mathbf{A}$ as faithfully as possible, while at the same time enable the utilization of efficient algorithms for the solution of systems $\mathbf{Mz} = \mathbf{r}$. This section will describe the construction of two classes of preconditioners with special structure from a given power grid by exploiting its spatial geometry. The first preconditioner (2D Fast Transform preconditioner) is perfectly applicable to power delivery networks with negligible via resistances, while the second preconditioner (3D Fast Transform preconditioner) is proposed in order to handle 3D power delivery networks with significant via resistances.

Both preconditioners are based on the following observations:

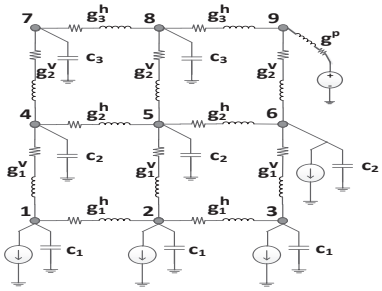
- Practical power grids are created as orthogonal wire meshes with very regular spatial geometries, with possibly some irregularities imposed by design constraints (e.g. some missing connections between adjacent nodes), and arranged in a few - typically 2 to 10 - metal layers of alternating routing directions (horizontal and vertical). Due to the presence of vias between successive metal layers, the actual grid has the structure of a 3D mesh, with very few planes along the third dimension.

- Almost all circuit elements (mainly resistances) in each metal layer have the same values, with few differences due to grid irregularities, as it is shown from data in [33].

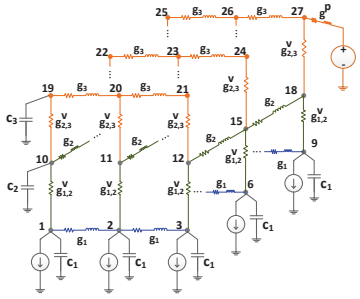
Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7



(i) Geometry structure of a 3D power grid with 3 layers. Vias represent connections between adjacent metal layers while the blue circle respresents the solder bump.



(ii) Regular grid obtained after the 2D regularization process.



(iii) Regular grid obtained after the 3D regularization process.

Figure 4-2: Example of a power delivery network with 3 horizontal and 3 vertical rails, along with the regular 2D and 3D grids used for preconditioning. The figure depicts only the VDD rails.

2D Fast Transform Preconditioner

The 2D Fast Transform preconditioner matrix that approximates the system matrix of the power grid by a process of regularization of the 3D power grid to a regular 2D grid, consisting of the following steps:

1. Determine the distinct x- and y-coordinates of all nodes in the different layers of the 3D grid, and take their Cartesian product to specify the location of the nodes in the regular 2D grid.

2. By disregarding via resistances between layers, collapse the 3D grid onto the regular 2D grid by adding together all horizontal branch conductances $g^h \equiv$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$\frac{1}{r^h+\frac{l^h}{h_k}}$ connected in parallel between adjacent nodes in the x-direction of the 2D grid, and all vertical branch conductances $g^v \equiv \frac{1}{r^v+\frac{l^v}{h_k}}$ connected in parallel between adjacent nodes in the y-direction of the 2D grid (where r^h, l^h denote the resistance and inductance of horizontal branches, and r^v, l^v denote the resistance and inductance of vertical branches - the inductances might not be present in the model). If a conductance of the 3D grid occupies multiple nodes of the regular 2D grid, it is decomposed into a corresponding number of pieces. The node capacitances corresponding to the same regular grid nodes are also added together during the collapsing.

3. In the regular 2D grid, substitute horizontal branch conductances by their average value in each horizontal rail, and vertical branch conductances by their average value in each horizontal slice (enclosed between two adjacent horizontal rails). Substitute node capacitances in each horizontal rail by their average value as well.

Fig. 4-2(i) depicts a 3D 3-layer power delivery network with $m = 3$ horizontal rails and $n = 3$ vertical rails in likewise-routed layers, while Fig 4-2(ii) shows the 2D regular grid that results from the previous regularization process used to construct the preconditioner matrix. If we use the depicted natural node numbering (proceeding horizontally, since this is always the routing direction of the lowest-level metal layer), the matrix $\mathbf{A}_{rl}(\mathbf{R}_b + \frac{\mathbf{L}_b}{h_k})^{-1}\mathbf{A}_{rl}^T + \frac{\mathbf{C}_n}{h_k}$ that corresponds to the regular 2D grid will be the following block-tridiagonal matrix:

$$\begin{bmatrix} \mathbf{T}_1 & -g_1^v\mathbf{I} & & \\ -g_1^v\mathbf{I} & \mathbf{T}_2 & -g_2^v\mathbf{I} & \\ & -g_2^v\mathbf{I} & \mathbf{T}_3 & -g_3^v\mathbf{I} \\ & & -g_3^v\mathbf{I} & \mathbf{T}_4 \end{bmatrix}$$

where $\mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3, \mathbf{T}_4$ are 3×3 tridiagonal matrices (each one corresponding to a horizontal rail of the 2D grid) which have the form:

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$$\mathbf{T}_1 = \begin{bmatrix} g_1^h+g_1^v+g^p+\frac{c_1}{h_k} & -g_1^h & \\ -g_1^h & 2g_1^h+g_1^v+\frac{c_1}{h_k} & -g_1^h \\ & -g_1^h & g_1^h+g_1^v+\frac{c_1}{h_k} \end{bmatrix}$$

$$\mathbf{T}_2 = \begin{bmatrix} g_2^h+g_1^v+g_2^v+\frac{c_2}{h_k} & -g_2^h & \\ -g_2^h & 2g_2^h+g_1^v+g_2^v+\frac{c_2}{h_k} & -g_2^h \\ & -g_2^h & g_2^h+g_1^v+g_2^v+\frac{c_2}{h_k} \end{bmatrix}$$

$$\mathbf{T}_3 = \begin{bmatrix} g_3^h+g_2^v+g_3^v+\frac{c_3}{h_k} & -g_3^h & \\ -g_3^h & 2g_3^h+g_2^v+g_3^v+\frac{c_3}{h_k} & -g_3^h \\ & -g_3^h & g_3^h+g_2^v+g_3^v+\frac{c_3}{h_k} \end{bmatrix}$$

$$\mathbf{T}_4 = \begin{bmatrix} g_4^h+g_3^v+\frac{c_4}{h_k} & -g_4^h & \\ -g_4^h & 2g_4^h+g_3^v+\frac{c_4}{h_k} & -g_4^h \\ & -g_4^h & g_4^h+g_3^v+\frac{c_4}{h_k} \end{bmatrix}$$

In the above, g_i^h is the average horizontal conductance in the i-th horizontal rail, g_i^v is the average vertical conductance in the i-th horizontal slice, and c_i is the average node capacitance in the i-th horizontal rail. Also h_k is the current analysis time-step (possibly variable), and $g^p \equiv \frac{1}{r^p+\frac{l^p}{h_k}}$ is the parasitic conductance of the supply pads (r^p and l^p denote the resistance and inductance of the supply pads respectively).

We observe that the form of the above matrix is almost identical to (4.6), with the exception of the pad parasitic conductance g^p in few places along the diagonal (considering that the number of voltage pads is much smaller than the number of nodes N). In order to obtain a preconditioner $\mathbf{M}$ with an exact form that can be efficiently solved by the application of a Fast Transform, we can just omit entirely those pad parasitics. However, we have found that in practice it is usually better to amortize the total sum of pad conductances of a specific horizontal rail (in the regular 2D grid) to all nodes of this rail, i.e. assume that all nodes of the i-th horizontal rail have pad conductance $\bar{g}_i^p = \frac{(\sum g^p)_i}{n}$, where $(\sum g^p)_i$ is the sum of the actual pad conductances attached to nodes of the i-th horizontal rail. This also has

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

the beneficial effect of making the preconditioner $\mathbf{M}$ non-singular in the case of DC analysis (where capacitances and inductances are absent). In the above example, the block $\mathbf{T}_1$ would become:

$$\mathbf{T}_1 = \begin{bmatrix} g_1^h+g_1^v+\bar{g}_1^p+\frac{c_1}{h_k} & -g_1^h & \\ -g_1^h & 2g_1^h+g_1^v+\bar{g}_1^p+\frac{c_1}{h_k} & -g_1^h \\ & -g_1^h & g_1^h+g_1^v+\bar{g}_1^p+\frac{c_1}{h_k} \end{bmatrix}$$

where $\bar{g}_1^p = \frac{g^p}{3}$. It is not difficult to generalize the procedure to an arbitrary $m \times n$ power grid. In that case, the preconditioner will comprise m blocks of size $n \times n$ and have the form (4.6), where $\alpha_i = g_i^h$, $\beta_i = g_i^v + g_{i-1}^v + \bar{g}_i^p + \frac{c_i}{h_k}$, $\gamma_i = -g_i^v$, $i = 1, \ldots, m$ (with $g_0^v = g_m^v = 0$).

3D Fast Transform Preconditioner

The above methodology for construction of a preconditioner neglects via resistances and approximates the 3D irregular power delivery network by a 2D regular structure. Neglecting via resistances in the preconditioner does not appear to be a very big problem in the analysis of moderately large designs (up to 1M nodes), since as it was reported in [17], they account for less than 1mV voltage drop in all the standard industrial power grid benchmarks [27]. However, after getting access to some even larger industrial grids, it became apparent that in those designs via resistances can be very significant and at least comparable to normal layer resistances. To illustrate this fact, Table 4.1 presents the maximum and average differences in voltage drop (or ground bounce) from the correct values when vias are neglected in those designs. The maximum error in voltage drop ranges from 0.5244V to 3.1626V for these particular grids (provided that sufficient supply voltage is available). As expected, neglecting via resistances in the construction of the preconditioner for those grids can considerably hinder the convergence of CG (and in fact, it has been observed that it can lead to divergence in some cases).

Based on the above observations, we extend the methodology presented in Section 4.4.1 to be applicable to large multi-layer power grid designs with signi-

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Table 4.1: Maximum and average voltage drop error when neglecting via resistances in the analysis of a set of large-scale industrial power grid benchmarks.

Benchmark	Size (# of nodes)	Max. Err. (V)	Avg. Err. (V)
ibm_y1000	0.42M	0.5244	0.0246
ibm_y800	0.65M	0.8000	0.0337
ibm_y600	1.17M	0.9315	0.0480
ibm_y500	1.68M	1.2194	0.0536
ibm_y400	2.62M	1.4904	0.0815
ibm_y300	4.69M	2.7567	0.1068
ibm_y250	6.72M	2.5914	0.1244
ibm_y200	10.51M	3.1626	0.2013

ficant via resistances, which cannot be handled by the 2D preconditioner. Given an irregular 3D power delivery network, we apply the following steps in order to formulate the 3D Fast Transform preconditioner:

1. Determine the distinct x- and y-coordinates of all nodes in the different layers of the given power grid, and take their Cartesian product to specify the location of the nodes in each layer of the regular 3D grid.

2. Substitute all branch conductances $g_i \equiv \frac{1}{r_i + \frac{l_i}{h_k}}$ in each metal layer by their average value within this layer (the inductances l_i might not be present in the model). Substitute node capacitances c_i in each layer by their average value as well. Finally, substitute the average value in all via conductances connecting two successive metal layers.

Fig 4-2(iii) shows the regular 3D grid that results from the aforementioned regularization process used to construct the preconditioner matrix when applied on the power delivery network of Fig. 4-1(i). If we use the depicted natural node numbering (proceeding horizontally in each layer, since this is always the routing direction of the lowest-level metal layer), the matrix $\mathbf{A}_{nb}(\mathbf{R}_b + \frac{\mathbf{L}_b}{h_k})^{-1}\mathbf{A}_{nb}^T + \frac{\mathbf{C}_n}{h_k}$ that corresponds to the regular 3D grid will be the following block-tridiagonal matrix:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & -g_{1,2}^v \mathbf{I}_{mn} & \\ -g_{1,2}^v \mathbf{I}_{mn} & \mathbf{M}_2 & -g_{2,3}^v \mathbf{I}_{mn} \\ & -g_{2,3}^v \mathbf{I}_{mn} & \mathbf{M}_3 \end{bmatrix}$$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

where

$$\mathbf{M}_1 = diag(\mathbf{T}_1, \mathbf{T}_1, \mathbf{T}_1), \quad \mathbf{M}_3 = diag(\mathbf{T}_3, \mathbf{T}_3, \mathbf{T}_3)$$

$$\mathbf{M}_2 = \begin{bmatrix} (g_2+g^v_{1,2}+g^v_{2,3}+\frac{c_2}{h_k})\mathbf{I}_n & -g_2\mathbf{I}_n & \\ -g_2\mathbf{I}_n & (2g_2+g^v_{1,2}+g^v_{2,3}+\frac{c_2}{h_k})\mathbf{I}_n & -g_2\mathbf{I}_n \\ & -g_2\mathbf{I}_n & (g_2+g^v_{1,2}+g^v_{2,3}+\frac{c_2}{h_k})\mathbf{I}_n \end{bmatrix}$$

$$\mathbf{T}_1 = \begin{bmatrix} g_1+g^v_{1,2}+\frac{c_1}{h_k} & -g_1 & \\ -g_1 & 2g_1+g^v_{1,2}+\frac{c_1}{h_k} & -g_1 \\ & -g_1 & g_1+g^v_{1,2}+\frac{c_1}{h_k} \end{bmatrix}$$

$$\mathbf{T}_3 = \begin{bmatrix} g_3+g^v_{2,3}+g^p+\frac{c_3}{h_k} & -g_3 & \\ -g_3 & 2g_3+g^v_{2,3}+\frac{c_3}{h_k} & -g_3 \\ & -g_3 & g_3+g^v_{2,3}+\frac{c_3}{h_k} \end{bmatrix}$$

In the above, g_i and c_i denote the average branch conductance and the average node capacitance in the i-th metal layer. Also h_k is the current analysis time-step (possibly variable), $g^v_{i,i+1}$ is the average via conductance connecting the i-th and (i+1)-th metal layers, and $g^p \equiv \frac{1}{r^p+\frac{l^p}{h_k}}$ is the parasitic conductance of the supply pads.

We observe that the form of the above matrix is almost identical to (4.13), with the exception of the pad parasitic conductance g^p in few places along the diagonal of the $\mathbf{M}_3$ block that corresponds to the uppermost metal layer. As in the 2D preconditioner construction, we obtain a preconditioner $\mathbf{M}$ with an exact form that can be efficiently solved by a 3D Fast Transform solver by amortizing the total sum of pad conductances of the uppermost metal layer (in the regular 3D grid) to all nodes of this layer, which also has the beneficial effect of making the preconditioner $\mathbf{M}$ non-singular in the case of DC analysis (where capacitances are absent). More specifically, we assume that all nodes of the uppermost layer have pad conductance $\bar{g}^p = \frac{\sum g^p}{nm}$, where $\sum g^p$ is the sum of the actual pad conductances attached to nodes of the uppermost layer. In the above example, the blocks $\mathbf{T}_3$ of $\mathbf{M}_3$ would become:

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

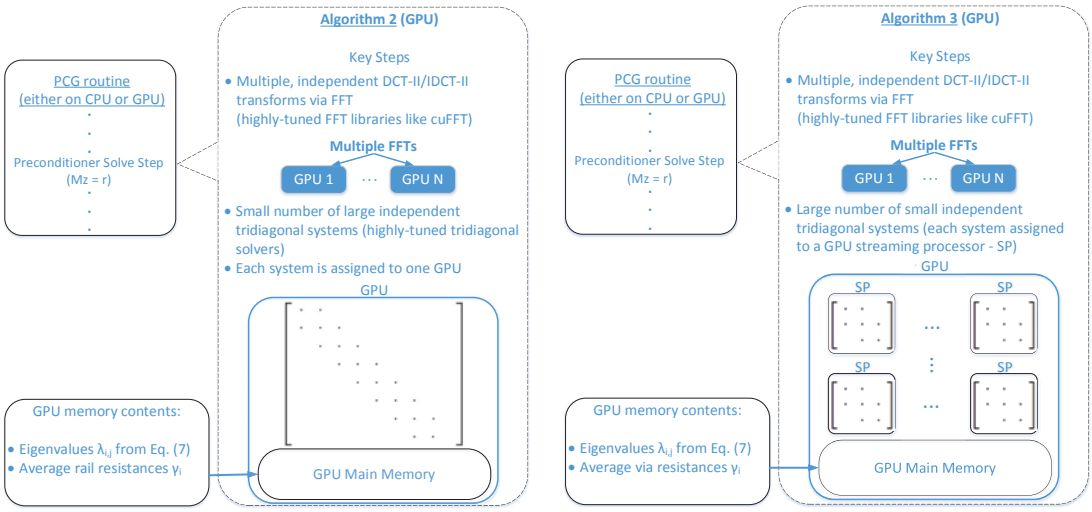
$$\mathbf{T}_3 = \begin{bmatrix} g_3+g_{2,3}^v+\bar{g}^p+\frac{c_3}{h_k} & -g_3 & \\ -g_3 & 2g_3+g_{2,3}^v+\bar{g}^p+\frac{c_3}{h_k} & -g_3 \\ & -g_3 & g_3+g_{2,3}^v+\bar{g}^p+\frac{c_3}{h_k} \end{bmatrix}$$

where $\bar{g}^p = \frac{g^p}{9}$. It is not difficult to generalize the procedure to an arbitrary power grid with m horizontal rails, n vertical rails and l layers. In that case, the preconditioner will comprise l blocks of size $mn \times mn$ and have the form (4.13), where $\alpha_i = g_i,\ (i = 1, \ldots, l)$, $\beta_i = g_{i-1,i}^v + g_{i,i+1}^v + \frac{c_i}{h_k},\ (i = 1, \ldots, l-1,$ with $g_{0,1}^v = 0)$, $\beta_l = g_{l-1,l}^v + \bar{g}^p + \frac{c_l}{h_k}$, and $\gamma_i = g_{i,i+1}^v\ (i = 1, \ldots, l-1)$.

Both the 2D and the 3D preconditioner construction requires only one parsing of the netlist of the electrical circuit representing the power grid, and is of complexity $\mathcal{O}(N)$ (considering that the number of electrical elements is of the same order as the number of nodes N), which is very inexpensive since it represents a one-time cost, roughly comparable to one iteration (and amortized over multiple iterations) of the PCG method. During transient analysis with variable time-step (which is almost always used in practical simulation scenarios - and completely rules out direct methods), the construction has to be repeated at every change of time-step in the same $\mathcal{O}(N)$ operations (this is also necessary for all other known preconditioners, and is in fact very expensive for some of them e.g. multigrid preconditioners). However, a considerable simplification is possible in the very common case of resistive or RC-only electrical models (i.e. when inductances are absent from the model) since the change of time-step does not affect any actions in the construction procedure, and thus there is no need for a full reconstruction (but only, actually, an update in the eigenvalues (4.8) of the preconditioner matrix).

Apart from the near-optimal complexity of solving the systems $\mathbf{Mz} = \mathbf{r}$, one other salient feature of the proposed preconditioners is that there is no need for explicit storage of the preconditioner matrix $\mathbf{M}$ from (4.6) or (4.13). As it is easily observed, only the eigenvalues (4.8) and the values γ_i of $\mathbf{M}$ are necessary in the execution of Algorithms 6 and 7. These are the only necessary values for formulation of the tridiagonal system (4.11) with coefficient matrix (4.12) and the tridiagonal

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7



(i) Key steps for the application of the preconditioner for 2D power grids.

(ii) Key steps for the application of the preconditioner for 3D power grids.

Figure 4-3: The key steps for the application of the proposed preconditioners and their mapping onto the available GPUs.

system (4.19) with coefficient matrix (4.20) that correspond to the 2D and the 3D preconditioning approaches respectively. Thus, only storage for those $mn + (m-1)$ values for the 2D algorithm and $lmn + (l-1)$ values for the 3D algorithm needs to be allocated. A small memory footprint is very important for mapping the algorithm onto architectures with limited available memory space such as GPUs.

4.4.2 Procedure Implementation and Opportunities for Parallelism

After the preconditioner construction and storage, the whole procedure involves execution of Algorithm 5 with Algorithm 6 for the 2D case or Algorithm 7 for the 3D case in place of the preconditioner solve step $\mathbf{Mz} = \mathbf{r}$. Every part of this procedure offers ample multi-grain parallelism, both data- and task-level, thus enabling highly parallel computing efficiency. This comes in contrast with most standard preconditioning methods, such as incomplete factorizations, which have limited parallelism, either data-level or task-level. The details for each major part of the procedure, as well as the opportunities for introducing parallelism are presented below.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Main CG Algorithm

As seen in Algorithm 5, apart from the preconditioner solve step, the PCG method involves 2 inner products and 1 sparse matrix-vector product per iteration, which can be implemented efficiently by available BLAS-1 and BLAS-2 (Basic Linear Algebra Subroutines) kernels. The algorithm also has 3 scalar-vector products with vector updates per iteration which can be fully parallelized.

DCT-II & IDCT-II Transforms

As already mentioned in Section 4.3.1, in order to apply the independent DCT-II and IDCT-II transforms of size n of Algorithm 6 (in steps 1 and 5) and of size n and m of Algorithm 7 (steps 1, 7 and 3, and 9 respectively) we can use a modification of the one-dimensional FFT algorithm [36], which gives a near-optimal sequential complexity ($\mathcal{O}(n \log n)$ for transforms of size n) for each of these transforms. FFT is also a highly parallel algorithm and an ideal candidate for mapping onto a multi-core processor or a GPU, with a parallel complexity of $\mathcal{O}((n \log n)/p)$, where p is the number of available processors [15]. This parallelization gain is especially evident on GPUs which offer a large amount of processing cores and can greatly reduce the cost of applying the one-dimensional FFT.

Solution of Tridiagonal Systems

One of the most time-consuming operations in the proposed algorithms is the solution of tridiagonal systems, either a small number of large ones (Algorithm 6) or a large number of small ones (Algorithm 7). However, the solution of tridiagonal systems offers abundant data-level parallelism as well, and various algorithms have been proposed in the literature for its implementation on parallel architectures. These can be classified to algorithms that target coarse-grain parallelism (and are appropriate for multi-core processors) such as two-way Gaussian elimination or Bondelli's algorithm [30], and to algorithms that exploit fine-grain parallelism (and are appropriate for GPUs) such as Parallel Cyclic Reduction [44] for systems with large dimensions or even the naive Thomas algorithm [35] for small tridiagonal

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

systems, where the latter two algorithms can offer the greatest speedup, as was naturally expected.

Task-Level Parallelism

The proposed algorithms entail a large number of one-dimensional DCT-II and IDCT-II transforms, and solution of tridiagonal linear systems. In each step, the operations are totally independent and thus each one can be solved separately from the others. This translates to additional task-level parallelism, which can lead to further acceleration of the whole preconditioner solve step in multi-GPU systems, where all independent transforms and tridiagonal solvers can be executed in parallel without requiring any data communication between different GPUs. Figure 4-3 depicts the main steps for mapping the proposed algorithms on a single- or multi-GPU system. As we can observe, the DCT-II/IDCT-II transforms can be executed in parallel either on a single or on multiple GPUs, without requiring any communication between the individual GPUs. In addition, the proposed algorithms can greatly benefit from the existence of multiple GPUs for the solution of the independent tridiagonal systems. In the 2D algorithm (Figure 4-3(i)) each tridiagonal system is mapped onto one of the available GPUs and solved independently by employing an off-the-shelf tridiagonal solution routine which exploits data-level parallelism within the algorithm. In the case of the 3D algorithm (Figure 4-3(ii)), the mapping onto a GPU can be made much more efficient by assigning each SP of the available GPU the task of solving one small tridiagonal system. Especially for $l = 2, 3$, analytical solutions can be provided for these systems, while for $l = 4, \ldots, 8$, we can utilize simple but effective tridiagonal solution algorithms such as the Thomas algorithm [35].

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

4.5 Experimental Results

4.5.1 Experimental Setup

To evaluate the performance of our methodology for power grid analysis, we compared four methods for solving the linear system (4.4i): CHOLMOD [12] which is a state-of-the-art CPU-based direct solver for sparse SPD linear systems, the PCG method with zero-fill Incomplete Cholesky preconditioner (ICCG), the PCG method with a support graph-based preconditioner (SPGCG) similar to the one presented in [46], and the proposed methods of using PCG with the Fast Transform preconditioners (FTCG-2D and FTCG-3D). We have ported CHOLMOD, ICCG, and FTCG on both a multi-core and a GPU architecture, while SPGCG was ported only on a multi-core architecture. For the GPU implementation of FTCG-2D and FTCG-3D, we have ported the entire CG iterative method (Algorithm 5) and the preconditioner solve step (Algorithm 6 and Algorithm 7) on the GPU. This eliminates the need for additional memory transfers between the host and the GPU and reduces the communication overhead, provided that the GPU has sufficient memory to accommodate the algorithm's working set (especially the system matrix $\mathbf{A}$ in sparse form). The only part of our algorithm that is implemented on the CPU for the GPU implementation is the preconditioner's initial construction and reconstruction procedure (during a time-step change). This procedure effectively results in the computation of the eigenvalues (4.8) and the values γ_i of the preconditioner matrix $\mathbf{M}$, which is the only information needed to store the preconditioner. Afterwards, the CPU is responsible for transferring these values to the main memory of the GPU (to update the elements of the preconditioner).

We have used Intel Math Kernel Library (MKL) [1] for implementing the CPU versions of the ICCG and FTCG-CPU algorithms, and CUDA library [2] (version 5.5, along with CUBLAS, CUSPARSE and CUFFT libraries) for mapping the FTCG algorithm on the GPU. Both the multi-core and the GPU implementation of FTCG involve the whole PCG method described in Algorithm 5, with Algorithm 6

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

or Algorithm 7 in place of the preconditioner solve step of line 6. We note that both MKL and CUDA libraries contain implementations of BLAS-1 and BLAS-2 kernels, FFT routines and tridiagonal solvers, all especially optimized for execution on multi-core and GPU architectures respectively. We executed all experiments on a Linux workstation, comprising an Intel Core i7 processor running at 2.4GHz (6 cores and 24GB main memory) and an NVIDIA Tesla C2075 GPU with 5GB of main memory. We used Intel and NVIDIA compilers for compiling our source code for the CPU and the GPU respectively, with the optimizations flags that resulted to the lowest execution time.

We have employed a set of industrial power grid designs [27] and a set of synthetic benchmarks, ranging from simple to more complicated designs, for the experimental validation of the proposed approach. The set of benchmarks include both benchmarks with negligible via resistances (ibmpg* and bench*_2D) as well as benchmarks with significant via resistances (ibm_y* - also used in research works [43] [13] - and bench*_3D). Table 4.2 and Table 4.3 present the details of each benchmark. The industrial (IBM) benchmarks are typical representatives of quite irregular designs from custom microprocessors, while the synthetic benchmarks are representatives of regular power delivery networks that are produced from most industrial automated routing tools. The synthetic benchmarks were full-RLC and had typical layer parameters (e.g. average resistance) taken from the industrial benchmarks and real designs. To compare the five different methods for power grid analysis, we have conducted transient simulation with variable time-step over a total of 1000 time-steps.

4.5.2 Transient Analysis Results for the Industrial Benchmarks

The simulation results for the power grid benchmarks with negligible and significant via resistances are presented in Table 4.4 and Table 4.5 respectively. Execution time (T) refers to the average time required for solution at each time-step, including

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Table 4.2: Circuit details and average number of iterations required for convergence of each iterative method over all simulation time-steps for the set of power grid 2D benchmarks with negligible via resistances. N is the total number of nodes, N_r is the number of resistors, N_l is the number of metal layers, and It denotes the number of iterations required for convergence by the corresponding method.

Benchmark	N	N_r	N_l	It_{ICCG}	It_{SPGCG}	It_{FTCG}
ibmpg2	127K	208K	5	207	80	38
ibmpg3	851K	1.40M	5	1457	128	306
ibmpg4	953K	1.56M	6	405	96	59
ibmpg5	1.07M	1.07M	3	1334	116	192
ibmpg6	1.67M	1.64M	3	1479	220	261
ibmpgnew1	1.46M	1.42M	N/A	1059	108	237
ibmpgnew2	1.46M	2.35M	N/A	1657	112	354
ibmX400	2.62M	2.62M	N/A	1657	112	354
bench1_2D	525K	1.04M	2	541	22	18
bench2_2D	4.05M	8.38M	4	1404	26	25
bench3_2D	6.29M	12.57M	5	1743	25	24

any overhead for matrix re-factorization (in CHOLMOD) and preconditioner reconstruction (in iterative methods) whenever the time-step changes. We have employed double-precision arithmetic for the benchmarks, while the iterative solvers were terminated when the solution residual was below 10^{-6}. This threshold is typically sufficient for ensuring a maximum error less than $1mV$ (and effectively results in perfect accuracy that is indistinguishable from direct methods).

As we can observe, CHOLMOD (the direct solver) suffers from a super-linear increase in execution time as the size of the power grid increases. In fact, the analysis of the largest designs was infeasible due to excessive memory requirements. On the other hand, ICCG, SPGCG, and PCG using the proposed Fast Transform-based preconditioners (FTCG) achieved lower execution times when implemented on a multi-core CPU (where CHOLMOD was also executed). In particular, the multi-core implementation of the FTCG method (FTCG-CPU) showed a speedup ranging from $12.79X$ to $77.85X$ in comparison to CHOLMOD.

Restricting now the comparison to the iterative methods, we observe a significant acceleration of the convergence rate (or reduction in the number of iterations) of

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Table 4.3: Circuit details and average number of iterations required for convergence of each iterative method over all simulation time-steps for the set of power grid 3D benchmarks with significant via resistances. N is the total number of nodes, N_r is the number of resistors, N_l is the number of metal layers, and It denotes the number of iterations required for convergence by the corresponding method.

Benchmark	N	N_r	N_l	It_{ICCG}	It_{SPGCG}	$It_{FTCG-3D}$
ibm_y1000	419K	601K	8	272	15	14
ibm_y800	655.8K	940.6K	8	302	17	14
ibm_y600	1.17M	1.68M	8	346	86	13
ibm_y500	1.68M	2.41M	8	369	470	13
ibm_y400	2.62M	3.77M	8	404	229	14
ibm_y300	4.68M	6.73M	8	436	373	16
ibm_y250	6.72M	9.66M	8	466	531	20
ibm_y200	10.5M	15.1M	8	466	372	18
bench1_3D	3.1M	6.0M	3	951	48	75
bench2_3D	16.8M	33.4M	8	1575	76	52
bench3_3D	18.9M	37.5M	6	1551	76	57
bench4_3D	22.2M	44.0M	7	1379	83	54

PCG when our Fast Transform preconditioner is applied. In particular, the proposed preconditioner was able to reduce the number of iterations by a factor ranging from $4.4X$ to $6.8X$ compared to the incomplete Cholesky preconditioner for the industrial benchmarks. This is a testament to the capability of the proposed preconditioner to provide an extremely good approximation of matrices of actual power grids. In addition, the number of iterations appears fairly constant with the increase of the problem size, while it exhibits a linear increase for the incomplete Cholesky preconditioner (which is a well-known theoretical fact [4]). In fact, for very regular problems (like the synthetic benchmarks) the constant number of iterations of Fast Transform preconditioners can be proven theoretically [10] [14]. Moreover, FTCG is able to greatly reduce the iteration count for the 3D industrial benchmarks when compared to SPGCG by a factor ranging from $1.07X$ to $36.15X$. The 3D industrial benchmarks are quite dense designs which can tax the efficiency of the support-graph based preconditioner. This is not the case for the 2D industrial designs, where the SPGCG preconditioner outperforms FTCG in terms of iterations count. However,

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Table 4.4: Runtime results for the four solvers on the 2D benchmarks. *Iter.* is the average number of iterations required for convergence of each iterative method over all simulation time-steps. T denotes the average time required for the solution at each time-step. Spd_{CHLMD}, Spd_{ICCG}, and Spd_{SPGCG} denote the speedup of $FTCG_{CPU}$ and $FTCG_{GPU}$ over $CHLMD_{CPU}$, $ICCG_{CPU}$, and $SPGCG_{CPU}$ respectively.

Benchmark	$CHLMD_{CPU}$	$ICCG_{CPU}$	$SPGCG_{CPU}$	$FTCG_{CPU}$				$FTCG_{GPU}$			
	$T(s)$	$T(s)$	$T(s)$	$T(s)$	Spd_{CHLMD}	Spd_{ICCG}	Spd_{SPGCG}	$T\ (s)$	Spd_{CHLMD}	Spd_{ICCG}	Spd_{SPGCG}
ibmpg2	4.9	1.73	1.08	0.33	14.96X	5.32X	3.3X	0.04	139.71X	49.66X	30.79X
ibmpg3	130.2	83.58	7.79	6.92	18.82X	12.09X	1.12X	0.57	226.35X	145.43X	13.53X
ibmpg4	221.9	21.36	10.2	2.85	77.85X	7.53X	3.57X	0.23	986.22X	95.45X	45.33X
ibmpg5	110.9	82.36	11.29	8.67	12.79X	9.51X	1.30X	0.58	189.32X	140.79X	19.27X
ibmpg6	235.1	143.56	23.65	11.96	19.65X	12.01X	1.97X	0.85	275.17X	168.27X	27.68X
ibmpgnew1	353.0	82.43	12.45	7.89	44.7X	10.46X	1.57X	0.53	661.56X	154.88X	23.33X
ibmpgnew2	339.9	172.38	18.80	15.39	22.08X	11.21X	1.22X	1.03	329.78X	167.47X	18.24X
ibmX400	553.3	1876	357.82	375	20.65	26.7X	17.3X	375	2.58	214.3X	138.7X
bench1_2D	67.3	26.7	3.66	1.8	37.61X	14.83X	2.03X	0.08	846.5X	333.7X	45.74X
bench2_2D	1523.7	542	28.99	21.4	71.34X	25.32X	1.35X	0.72	2119.7X	752.5X	40.25X
bench3_2D	1631.1	1002	41.83	30.8	52.96X	32.53X	1.36X	1.11	1469.45	899.7X	37.55X

Table 4.5: Runtime results for the four solvers on the 3D benchmarks. Spd_{CHLMD}, Spd_{ICCG}, and Spd_{SPGCG} denote the speedup of $FTCG{-}3D_{CPU}$ and $FTCG{-}3D_{GPU}$ over $CHLMD_{GPU}$, $ICCG_{GPU}$, and $SPGCG_{CPU}$ respectively.

Benchmark	$CHLMD_{GPU}$	$ICCG_{GPU}$	$SPGCG_{CPU}$	$FTCG-3D_{CPU}$				$FTCG-3D_{GPU}$			
	$T(s)$	$T(s)$	$T(s)$	$T(s)$	Spd_{CHLMD}	Spd_{ICCG}	Spd_{SPGCG}	$T\ (s)$	Spd_{CHLMD}	Spd_{ICCG}	Spd_{SPGCG}
ibm_y1000	15.85	2.87	0.46	0.55	28.72X	5.20X	0.82X	0.10	157.73X	28.60X	4.54X
ibm_y800	25.95	5.60	1.47	0.71	36.65X	7.90X	2.07X	0.15	176.51X	38.06X	9.98X
ibm_y600	50.94	11.52	8.47	1.20	42.45X	9.6X	7.05X	0.20	249.70X	56.47X	41.5X
ibm_y500	81.45	17.61	109.434	2.39	34.07X	7.37X	45.77X	0.34	240.26X	51.94X	322.81X
ibm_y400	123.615	30.99	61.40	3.15	39.24X	9.84X	19.5X	0.43	290.17X	72.74X	144.14X
ibm_y300	No mem.	61.50	259.16	6.60	N/A	9.33X	39.26X	0.85	N/A	72.18X	304.17X
ibm_y250	No mem.	95.91	525.34	12.45	N/A	7.70X	42.19X	1.94	N/A	49.33X	270.23X
ibm_y200	No mem.	151.71	583.26	15.72	N/A	9.65X	37.10X	2.01	N/A	75.47X	290.18X
bench1_3D	3476	168.84	49. 35	39.6	87.78X	4.26X	1.24X	2.1	551.74X	26.8X	23.5X
bench2_3D	No mem.	1463.98	221.45	124.56	N/A	11.75X	1.77X	9.57	N/A	50.99X	23.14X
bench3_3D	No mem.	1607.87	249.4	158.08	N/A	10.17X	1.57X	12.45	N/A	43.04X	20.03X
bench4_3D	No mem.	1686.16	304.79	172.06	N/A	9.79X	1.77X	14.49	N/A	38.78X	21.03X

as will be discussed, this is not depicted in the execution time, mainly due to the limited parallelism in the preconditioner solve step that SPGCG exhibits.

The reduced iteration count is the major factor for the speedup of the multi-core implementation of FTCG with respect to ICCG, which ranges from $5.32X$ to $12.09X$ and from $5.2X$ to $9.84X$ for the 2D and the 3D industrial benchmarks respectively. On the other hand, the inherent parallelism is the main acceleration factor of FTCG when compared with SPGCG. FTCG is able to achieve an acceleration factor ranging from $1.12X$ to $3.57X$ and from $2.07X$ to $45.77X$ for the 2-D and the 3D industrial benchmarks respectively. In addition, due to the complexity of

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

benchmark ibmX400, SPGCG was not able to simulate the corresponding power delivery network. The only benchmark where SPGCG outperforms FTCG is ibm_y1000, which is a rather small design and the direct solver that is employed in the preconditioner solve step of SPGCG is faster than the solution procedure of FTCG.

Additional acceleration is gained when GPUs are utilized. GPUs can take advantage of the inherent parallelism of FFT and the tridiagonal solution algorithm by employing their vast amount of computational resources. Our algorithms achieve an average speedup of $131.7X$ over ICCG for the 2D and an average speedup of $55X$ for the 3D industrial benchmarks. The acceleration becomes even more pronounced as the size of the power grid increases. FTCG achieves a speedup equal to $167.47X$ and $75.47X$ for a 1.46M-node 2D and a 10.5M-node 3D industrial benchmark respectively. FTCG also outperforms SPGCG when GPUs are employed. It achieves an average speedup of $22.85X$ and of $173.4X$ over SPGCG for the 2D and 3D industrial benchmarks respectively, while it achieves a speedup of $18.24X$ and of $290.18X$ for the largest 2D and 3D benchmark respectively. Although SPGCG has not been ported on a GPU, the direct solution algorithm that is employed in the corresponding preconditioner solve step entails limited parallelism. Even on a GPUs, the maximum speedup would be $2X$-$3X$, which still renders our algorithm superior.

4.5.3 Transient Analysis Results for the Synthetic Benchmarks

In order to further exemplify the robustness and efficiency of the proposed preconditioning algorithms, we have employed a set of synthetic benchmarks, with sizes ranging from 525K-nodes to 6.29M-nodes for the 2D case and from 3.1M-nodes to 22.2M-nodes for the 3D case. The corresponding simulation results are presented in Table 4.4 and Table 4.5.

CHOLMOD (the direct solver) suffers from a super-linear increase in memory consumption as the size of the power grid increases. As a result, it was not able to

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

simulate the largest 2D synthetic power grid and it was able to simulate only the smallest of the 3D designs. In terms of execution time, the multi-core implementation of FTCG achieved a speedup of $52.96X$ and $87.78X$ over the largest 2D and 3D power delivery network that CHOLMOD was able to simulate, while the GPU implementation of FTCG achieved a speedup of $1469.45X$ and $551.74X$ respectively.

As shown in Table 4.4 and Table 4.5, the multi-core implementation of FTCG outperforms both the multi-core and the GPU implementation of ICCG. FTCG achieves a speedup ranging from $14.83X$ to $32.53X$ for the 2D designs and a speedup ranging from $4.26X$ to $11.75X$ for the 3D designs. The efficiency of our algorithms is further increased when ported on a GPU, where they achieve a speedup ranging from $26.8X$ to $899.7X$ compared to ICCG. On the other hand, the multi-core implementation of SPGCG was able to achieve comparable number of iterations and execution time with FTCG both on the 2D and the 3D synthetic benchmarks. However, the GPU implementations of the proposed algorithms were able to achieve speedups ranging from $37.55X$ to $55.21X$ for the 2D synthetic benchmarks and from $16.37X$ to $29.83X$ for the 3D synthetic benchmarks. This is a testament for the high degree of parallelism that the proposed algorithms entail, which allows harnessing the computational capabilities of massively parallel architectures. Moreover, owing to the task-level parallelism further acceleration can be achieved by porting the proposed algorithms on a multi-GPU system (although this has not implemented in this work). This comes in contrast to the majority of power delivery simulation algorithms that provide limited parallelism.

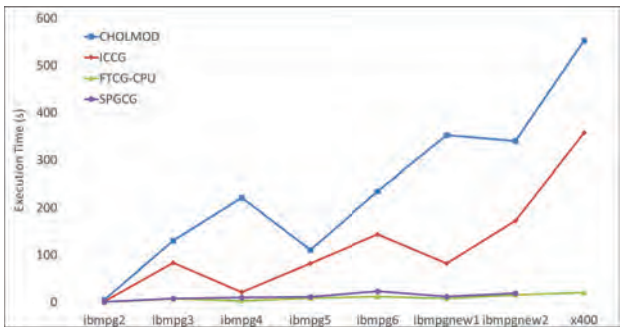
It is noted that both for the 2D and the 3D benchmarks the GPU execution time includes the communication overhead for transferring the updated eigenvalues (4.8) and the values γ_i of the preconditioner matrix after every reconstruction from the host to the GPU. However, the bandwidth of the PCI-Express bus was able to effectively hide this additional overhead. As a result, the time required for these data transfers was smaller than 1% of the execution time for each time-step. Moreover, the reconstruction step can be executed asynchronously with the execution of the

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

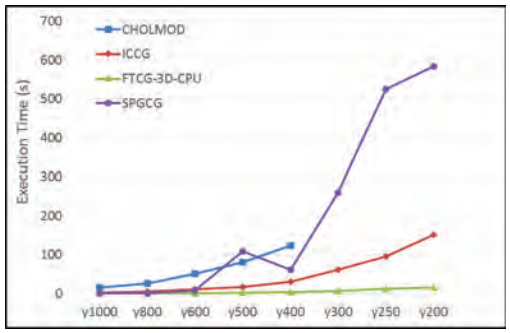
algorithm on the GPU and practically eliminate this communication overhead.

4.5.4 Scalability of FTCG and FTCG-3D

Fig. 4-4(i) and Fig. 4-4(ii) shows the runtime of the four solution algorithms under comparison for the 2D and the 3D industrial benchmarks respectively. Owing to the efficiency of their preconditioning mechanisms, both FTCG and FTCG-3D are able to achieve a favorable scaling with the size of the power delivery network. This is not the case for the other algorithms under comparison, and especially for the 3D case that exhibits a greater complexity, CHOLMOD, ICCG, and SPGCG exhibit a super-linear scaling. This is mainly due to the inefficiency of the preconditioning mechanisms that are use in these methods. The Incomplete Cholesky preconditioner used in the ICCG method is a general one and is not able to reduce the iteration count to a great extent, while the preconditioner of SPGCG is not able to handle dense graphs with increased complexity, thus increasing the time required for the solution of the preconditioner solve step.



(i) Runtime scalability of FTCG



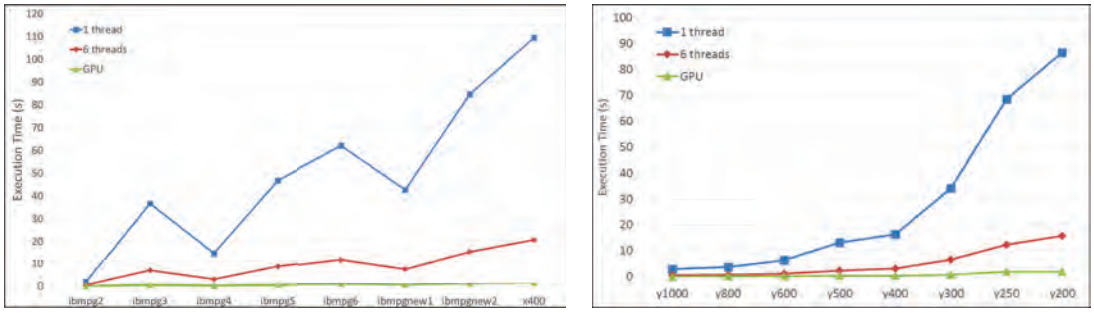
(ii) Runtime scalability of FTCG-3D

Figure 4-4: Runtime scalability of FTCG and FTCG-3D on the set of the industrial benchmarks.

In addition, the second important feature of FTCG and FTCG-3D is that they entail a large degree of parallelism, which allows for efficient harnessing of the computational resources of multi-core and massively parallel architectures. As shown in Fig. 4-5(i) and Fig. 4-5(ii) that depict the scalability for 1 and 6 threads as well as the GPU implementation of FTCG and FTCG-3D respectively, we can observe that

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

the proposed algorithms achieve a linear acceleration with the increasing number of threads. Additional speedup can be achieved with porting the algorithms on the GPU that offers a larger amount of computational resources. The speedup that is achieved in the parallel implementations for the largest 2D benchmark compared to the multi-core implementation using 1 thread is $5.3X$ and $84.8X$ for the multi-core implementation using 6 threads and the GPU implementation respectively, while the speedup that is achieved for the largest 3D benchmarks is $5.5X$ and $43.01X$ for the multi-core implementation using 6 threads and the GPU implementation respectively.



(i) Thread scalability of FTCG (ii) Thread scalability of FTCG-3D

Figure 4-5: Thread scalability of the 1-thread, 6-thread, and the GPU implementations of FTCG and FTCG-3D on the set of the industrial benchmarks.

4.5.5 Memory Efficiency

A salient feature of the proposed Fast Transform preconditioners is the elimination of the need for explicit storage of the preconditioner matrices $\mathbf{M}$, since only the eigenvalues from (4.8) and the values γ_i (that correspond to the average via resistance) need to be kept in memory. This matrix-less formulation of the proposed preconditioning approaches contributes significantly in the reduction of their memory requirements. We studied the memory efficiency of the proposed approaches by measuring the amount of memory required by FTCG, CHOLMOD, and ICCG for the analysis of the three largest benchmarks (including both industrial and synthetic ones). The corresponding results are presented in Table 4.6.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

We can observe that there is a dramatic increase in memory usage for the CHOLMOD algorithm with the increase of the grid size, where for the largest design CHOLMOD required $14.5GB$ of main memory. On the other hand, our algorithm provides scalable memory performance. The total amount of required memory increased linearly with the increase in the number of power grid nodes, while the algorithm consumed less memory by an average factor of $17.7X$ and $1.33X$ than CHOLMOD and ICCG respectively. Compared with ICCG, our method requires limited additional storage for the preconditioner matrix, thus memory requirements are further reduced. Even for our largest benchmark, the algorithm consumed less than $690MB$ of memory, which renders feasible the analysis of large-scale power grids on GPUs (which are characterized by the limited amount of main memory). In addition, limited memory consumption can have beneficial effects on the actual execution of our algorithm when it is mapped onto a GPU. In that case, the available memory can accommodate the algorithm's working set, thus eliminating the need for additional data transfers between the GPU and the host.

Table 4.6: Memory requirements (MB) for the largest 2D benchmarks of FTCG, CHOLMOD, and ICCG. $M_{FTCG-GPU}$ is the cumulative memory on the GPU and the CPU required by the FTCG-GPU implementation.

Benchmark	$M_{CHOLMOD}$	M_{ICCG}	$M_{FTCG-CPU}$	$M_{FTCG-GPU}$
ibmX400	3350	311	239	239
bench2_2D	7700	615	471	471
bench3_2D	14500	919	685	685

4.5.6 Efficiency Under Grid Irregularity

Another aspect that must be considered during the application of the proposed approaches is the degree to which the given power grid can deviate from regularity, since the proposed preconditioners are constructed by averaging branch resistances and node capacitances. In order to study the effect of grid irregularity on the preconditioners' behavior, we modified the 3.1M-node 3D synthetic benchmark

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

bench1_3D to have random values of branch conductances with varying degrees of randomness around their nominal values. Table 4.7 presents the average number of iterations required for convergence of FTCG-3D over multiple time-steps of transient analysis, after applying successively higher degree of irregularity ranging from 0.1% to 100%. As we can observe, FTCG-3D was extremely insensitive to the irregularity of the power grid, since even for an irregularity degree as high as ±75.0%, the number of iterations has increased very slightly compared to the regular grid. On the other hand, the number of iterations increases dramatically after an irregularity degree of ±85.0%. Although power delivery networks are usually highly regular and we do not expect such degrees of irregularity, these observations are a testament to the efficiency of the proposed preconditioning mechanisms for power grid analysis.

Table 4.7: Results of FTCG-3D under varying grid irregularity for the bench1_3D benchmark.

Branch conductance irregularity	Iterations	Execution Time (s)
0.0 %	75	2.1
± 0.1 %	78	2.2
± 0.5 %	80	2.2
± 1.0 %	82	2.3
± 1.5 %	84	2.4
± 2.0 %	84	2.4
± 5.0 %	88	2.5
± 10.0 %	90	2.5
± 20.0 %	93	2.6
± 50.0 %	95	2.6
± 75.0 %	98	2.6
± 85.0 %	196	5.5
± 100.0 %	679	19.1

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Chapter 5

Power Grid Electro-Thermal Simulation

The increased power density combined with the lower power supply voltages due to shrinking sizes, has led to a significant increase in current density. Larger current densities result in greater IR drop, while Joule heating (self-heating) effect becomes of critical importance and should be seriously taken into account [5] as it contributes to temperature rise and affects reliability. As the electrical resistivity is temperature-dependent, temperature variations on the power delivery network substantially modify the interconnect resistances contributing to the IR drop in the power grid. However, such effects are usually neglected during IR drop analysis due to the immense growing size of modern power delivery networks and the corresponding demands in computational resources for their analysis. Power grids can be extremely large, demanding abundant computational resources, while at the same time, thermal analysis requires even more resources in terms of speed and memory.

As IR drop on the power grid and temperature rise on the interconnects constitute undesirable issues with respect to the performance, reliability and functionality of nano-scale technology, extensive research has been performed to separately analyze each of these issues. However, few approaches have been suggested for a thermal-aware IR drop analysis, despite the fact that IR drop and temperature are directly

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

interdependent. Authors in [39] present an electrical-thermal co-simulation method, including Joule heating, air convection and fluidic cooling effects using the finite volume method with non-uniform rectangular grid. In the same context, a thermal-aware IR drop analysis for power grids is proposed in [47]. Both methods present the idea of iterative electrical-thermal simulations, updating the electrical resistivities in each iteration, including this way the thermal effects on each new electrical simulation. However, they focus on modeling aspects rather on simulation approaches.

The sheer size of power delivery networks forced researchers to focus on parallel architectures as a promising alternative for accelerating simulation algorithms, owing to their vast computational resources. Approach [24] presents a method for full-chip thermal analysis on GPUs. It formulates the problem of thermal analysis as a non-SPD problem and combines the GMRES method with an approximate inverse preconditioner for its solution. However, the proposed methodology does not consider a combined electro-thermal analysis approach. Furthermore, it contains limited potential for parallelism and the maximum achieved speedup is $2.24X$ when GPUs are utilized.

In this chapter, we present a new efficient and highly-parallel algorithm for electrical-thermal co-simulation for large-scale power grids, found in most contemporary nano-scale ICs. Our method combines the methodology for electrical analysis presented in Sec. 4.4.1 and a new preconditioning methodology for thermal simulation. In contrast to all previous approaches, it takes into account the basic thermal factors that contribute to temperature rises on the power grid such as Joule heating and heat from both the substrate and interconnects, while at the same time exhibits great potential of parallelism. We propose an efficient and highly-parallel preconditioning mechanism based on the application of a Fast Transform solver, which can be used in conjunction with a state-of-the-art iterative solution method in order to accelerate electrical-thermal analysis of power delivery networks. The benefits of the proposed method are twofold: i) the proposed preconditioning mechanism can accelerate the convergence rate of the iterative solution method by greatly reducing the required number of iterations, and ii) from a computational point of

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

view, it exhibits near-optimal computational complexity, low memory requirements, and great potential for parallelism, which can harness the computational power of parallel architectures, such as multi-core processors or GPUs, thus further reducing the amount of time required for simulation.

5.1 Methodology Overview

Due to the temperature dependence of resistance and Joule self-heating of the conductors, the electrical and thermal characteristics of the power delivery network form a nonlinear system of equations. In order to capture the effect of the thermal profile of the power delivery network, we follow a combined electrical-thermal simulation for the power delivery network. Having set the initial input as well as the boundary conditions for the thermal simulation, the first step is the electrical analysis of the power grid. The power model provides the current and power distribution profiles to the thermal model. Subsequently, the thermal model estimates the temperature profile and once electrical resistivities have been updated, they are forwarded as a new input to the power model. After a number of iterations, power and temperature calculations converge and the thermal-aware power grid analysis is terminated. Fig. 5-1 depicts the flow diagram of the proposed approach.

Electrical modeling of the power grid is performed as was described in Section 4.1. On the other hand, thermal modeling requires a different approach. Without loss of generality, we focus on steady-state thermal analysis. Steady-state thermal analysis aims at determining the temperature distribution within a chip given a power density distribution that does not change with time and amounts to solving Poisson's equation:

$$q(r) = -k_t \nabla^2 T(r) \tag{5.1}$$

where r is the spatial coordinate of the point at which temperature is being determined, $q(r)$ is the rate of heat flow, T is the temperature, and k_t the thermal conductivity of the material.

By discretizing (5.1) using the Finite Difference Method (FDM), we obtain the

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

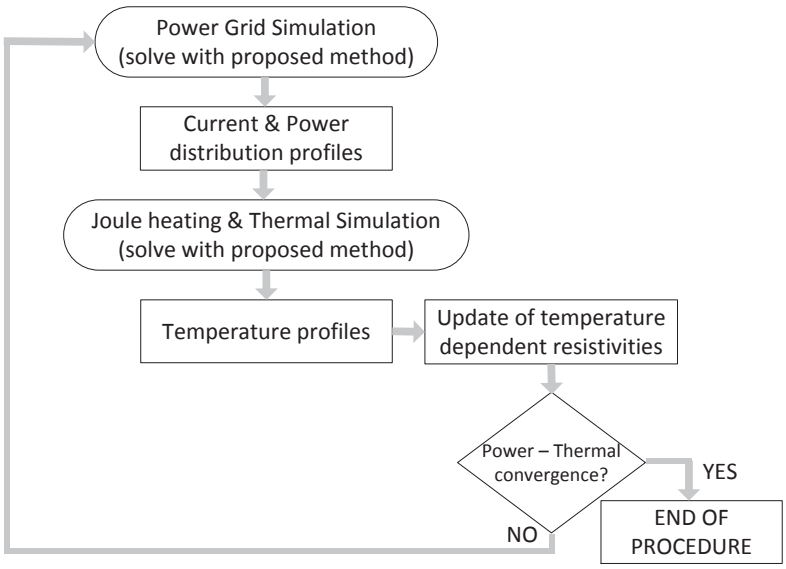


Figure 5-1: Electrical-Thermal positive feedback simulation loop.

following system of linear equations:

$$\mathbf{Gt} = \mathbf{p} \tag{5.2}$$

where $\mathbf{t}$ is the temperature vector at each point and $\mathbf{p}$ is the vector containing the total power generated within each element. Due to the electrical-thermal duality, each node in the discretization corresponds to a node in the circuit. Using the MNA formulation, matrix $\mathbf{G}$ contains the thermal resistors G_x, G_y, and G_z in the x, y, and z direction respectively, which can be computed as follows:

$$G_x = \frac{k_t(\Delta_y \cdot \Delta_z)}{\Delta_x},\ G_y = \frac{k_t(\Delta_x \cdot \Delta_z)}{\Delta_y},\ G_z = \frac{k_t(\Delta_x \cdot \Delta_y)}{\Delta_z}$$

where $\Delta_{\{x,y,z\}}$ is the length of the rectangle in each dimension. As we can observe, steps 1 and 2 respectively of the combined electro-thermal approach require the solution of a linear system. In order to accelerate the analysis procedure and enable analysis of large-scale power delivery networks, we utilize the PCG method (as both matrices can be shown to be symmetric and positive definite) for the solution of both systems. For the electrical analysis step, we apply the methodology presented

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

in Chapter 4. On the other hand, for thermal analysis we extend the 3D Fast Transform methodology presented in Section 4.3.2 in order to be applicable to full 3D networks that are produced from (5.2), and we apply the full 3D Fast Transform solver as a preconditioner inside an iterative method.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

5.2 Fast Transform Solvers for Full 3D Networks

Let $\mathbf{M}$ be a $N \times N$ block-tridiagonal matrix with l blocks of size $mn \times mn$ each (overall $N = lmn$) with the following form:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & -\delta_1\mathbf{I}_{mn} & & & \\ -\delta_1\mathbf{I}_{mn} & \mathbf{M}_2 & -\delta_2\mathbf{I}_{mn} & & \\ & \cdot & \cdot & \cdot & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \mathbf{M}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \mathbf{M}_l \end{bmatrix} \tag{5.3}$$

where $\mathbf{I}_{mn}$ is the $mn \times mn$ identity matrix, $\mathbf{M}_i$, $i = 1, \ldots, l$, are themselves $mn \times mn$ block-tridiagonal matrices of the form:

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{T}_i + \gamma_i\mathbf{I}_n & -\gamma_i\mathbf{I}_n & & & \\ -\gamma_i\mathbf{I}_n & \mathbf{T}_i + 2\gamma_i\mathbf{I}_n & -\gamma_i\mathbf{I}_n & & \\ & \cdot & \cdot & \cdot & \\ & & -\gamma_i\mathbf{I}_n & \mathbf{T}_i + 2\gamma_i\mathbf{I}_n & -\gamma_i\mathbf{I}_n \\ & & & -\gamma_i\mathbf{I}_n & \mathbf{T}_i + \gamma_i\mathbf{I}_n \end{bmatrix}$$

where $\mathbf{I}_n$ is the $n \times n$ identity matrix and $\mathbf{T}_i$ have the form (4.7). Thus, the eigenvectors of the diagonal blocks of $\mathbf{M}_i$ are the same as those of $\mathbf{T}_i$, with values given by (4.9). If $\mathbf{Q}_n = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ denotes the matrix whose columns are the eigenvectors $\mathbf{q}_j$, then due to the eigen-decomposition of $\mathbf{T}_i$, $i = 1, 3, ..., n$ we have $\mathbf{Q}_n^T\mathbf{T}_i\mathbf{Q}_n = \mathbf{\Lambda}_i = diag(\lambda_{i,1}, \ldots, \lambda_{i,n})$, where $\mathbf{\Lambda}_i$ is the diagonal matrix with the eigenvalues of each $\mathbf{T}_i$ in its main diagonal. By exploiting this diagonalization of matrices $\mathbf{T}_i$, system $\mathbf{Mz} = \mathbf{r}$ with $\mathbf{M}$ of the form (5.3) is equivalent to the following system (due to $\mathbf{Q}_n^T\mathbf{Q}_n = \mathbf{I}$):

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$$\begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{M} \begin{bmatrix} \mathbf{Q}_n & & \\ & \ddots & \\ & & \mathbf{Q}_n \end{bmatrix} \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{z}$$
$$= \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{r} \Leftrightarrow$$

$$\begin{bmatrix} \tilde{\mathbf{M}}_1 & -\delta_1\mathbf{I}_{mn} & & & & \\ -\delta_1\mathbf{I}_{mn} & \tilde{\mathbf{M}}_2 & -\delta_2\mathbf{I}_{mn} & & & \\ & \cdot & \cdot & \cdot & & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \tilde{\mathbf{M}}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} & \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \tilde{\mathbf{M}}_l & \end{bmatrix} \tilde{\mathbf{z}} = \tilde{\mathbf{r}} \qquad (5.4)$$

where:

$$\tilde{\mathbf{M}}_i = \begin{bmatrix} \mathbf{\Lambda}_i^{(1)} & -\gamma_i\mathbf{I}_n & & & \\ -\gamma_i\mathbf{I}_n & \mathbf{\Lambda}_i^{(2)} & -\gamma_i\mathbf{I}_n & & \\ & \cdot & \cdot & \cdot & \\ & & -\gamma_i\mathbf{I}_n & \mathbf{\Lambda}_i^{(2)} & -\gamma_i\mathbf{I}_n \\ & & & -\gamma_i\mathbf{I}_n & \mathbf{\Lambda}_i^{(1)} \end{bmatrix}$$

$$\tilde{\mathbf{z}} = \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{z}, \quad \tilde{\mathbf{r}} = \begin{bmatrix} \mathbf{Q}_n^T & & \\ & \ddots & \\ & & \mathbf{Q}_n^T \end{bmatrix} \mathbf{r},$$

and $\mathbf{\Lambda}_i^{(1)} = diag(\lambda_{i,1}^{(1)}, \ldots, \lambda_{i,n}^{(1)})$, $\mathbf{\Lambda}_i^{(2)} = diag(\lambda_{i,1}^{(2)}, \ldots, \lambda_{i,n}^{(2)})$ are diagonal matrices with the eigenvalues of $\mathbf{T}_i + \gamma_i\mathbf{I}_n$, $\mathbf{T}_i + 2\gamma_i\mathbf{I}_n$, which are the following:

$$\lambda_{i,j}^{(1)} = \gamma_i + \beta_i + \alpha_i(2\cos\frac{(j-1)\pi}{n} - 2),\ j = 1, \ldots, n$$
$$\lambda_{i,j}^{(2)} = 2\gamma_i + \beta_i + \alpha_i(2\cos\frac{(j-1)\pi}{n} - 2),\ j = 1, \ldots, n$$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

If $\mathbf{P}$ is again the $mn \times mn$ permutation matrix that reorders the elements of a vector or the rows of a matrix as $1, n+1, \ldots, (m-1)n+1,\ 2, n+2, \ldots, (m-1)n+2,\ \ldots,\ n, n+n, \ldots, (m-1)n+n$, and $\mathbf{P_1}$, $\mathbf{P_1^T}$ denote the block-diagonal $lmn \times lmn$ permutation matrices $\mathbf{P_1} = diag(\mathbf{P}, \ldots, \mathbf{P})$, $\mathbf{P_1^T} = diag(\mathbf{P}^T, \ldots, \mathbf{P}^T)$, then the system (5.4) is further equivalent to:

$$\begin{bmatrix} \mathbf{D}_1 & -\delta_1\mathbf{I}_{mn} & & & \\ -\delta_1\mathbf{I}_{mn} & \mathbf{D}_2 & -\delta_2\mathbf{I}_{mn} & & \\ & \cdot & \cdot & \cdot & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \mathbf{D}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \mathbf{D}_l \end{bmatrix} \tilde{\mathbf{z}}^{\mathbf{P}_1} = \tilde{\mathbf{r}}^{\mathbf{P}_1} \tag{5.5}$$

where $\mathbf{D}_i = diag(\tilde{\mathbf{T}}_{i,1}, \ldots, \tilde{\mathbf{T}}_{i,n}),\ i = 1, \ldots, l$, with $\tilde{\mathbf{T}}_{i,j},\ j = 1, \ldots, n$ being $m \times m$ tridiagonal matrices of the form:

$$\tilde{\mathbf{T}}_{i,j} = \begin{bmatrix} \lambda_{i,j}^{(1)} & -\gamma_i & & & \\ -\gamma_i & \lambda_{i,j}^{(2)} & -\gamma_i & & \\ & \cdot & \cdot & \cdot & \\ & & -\gamma_i & \lambda_{i,j}^{(2)} & -\gamma_i \\ & & & -\gamma_i & \lambda_{i,j}^{(1)} \end{bmatrix}$$

$$= \gamma_i \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \cdot & \cdot & \cdot & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} + (\alpha_i(2\cos\frac{(j-1)\pi}{n} - 2) + \beta_i)\mathbf{I}_m$$

and $\tilde{\mathbf{z}}^{\mathbf{P}_1} = \mathbf{P_1}\tilde{\mathbf{z}}, \tilde{\mathbf{r}}^{\mathbf{P}_1} = \mathbf{P}_1\tilde{\mathbf{r}}$. If $\tilde{\mathbf{\Lambda}}_{i,j} = diag(\tilde{\lambda}_{i,j,1}, \ldots, \tilde{\lambda}_{i,j,m})$ is the diagonal matrix with the eigenvalues of $\tilde{\mathbf{T}}_{i,j}$, which are:

$$\tilde{\lambda}_{i,j,k} = \gamma_i(2\cos\frac{(k-1)\pi}{m} - 2) + \alpha_i(2\cos\frac{(j-1)\pi}{n} - 2) + \beta_i,\ k = 1, \ldots, m \tag{5.6}$$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

and $\mathbf{Q}_m$ is the common matrix of eigenvectors for all $\tilde{\mathbf{T}}_{i,j}$, then again by similar reasoning as in (5.4), the system (5.5) is equivalent to:

$$\begin{bmatrix} \tilde{\mathbf{D}}_1 & -\delta_1\mathbf{I}_{mn} & & & & \\ -\delta_1\mathbf{I}_{mn} & \tilde{\mathbf{D}}_2 & -\delta_2\mathbf{I}_{mn} & & & \\ & \cdot & \cdot & \cdot & & \\ & & -\delta_{l-2}\mathbf{I}_{mn} & \tilde{\mathbf{D}}_{l-1} & -\delta_{l-1}\mathbf{I}_{mn} & \\ & & & -\delta_{l-1}\mathbf{I}_{mn} & \tilde{\mathbf{D}}_l & \end{bmatrix} \tilde{\tilde{\mathbf{z}}} = \tilde{\tilde{\mathbf{r}}} \tag{5.7}$$

where $\tilde{\mathbf{D}}_i = diag(\tilde{\mathbf{\Lambda}}_{i,1}, \ldots, \tilde{\mathbf{\Lambda}}_{i,n})$ and

$$\tilde{\tilde{\mathbf{z}}} = \begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \tilde{\mathbf{z}}^{\mathbf{P}_1}, \quad \tilde{\tilde{\mathbf{r}}} = \begin{bmatrix} \mathbf{Q}_m^T & & \\ & \ddots & \\ & & \mathbf{Q}_m^T \end{bmatrix} \tilde{\mathbf{r}}^{\mathbf{P}_1}$$

If now $\mathbf{P}_2$ is a permutation matrix of size $N \times N$ that reorders the elements of a vector or the rows of a matrix as $1, mn+1, 2mn+1, \ldots, (l-1)mn+1,\ 2, mn+2, 2mn+2, \ldots, (l-1)mn+2,\ \ldots,\ mn, mn+mn, 2mn+mn, \ldots, (l-1)mn+mn$, and $\mathbf{P}_2^T$ is the inverse permutation matrix, then system (5.7) is equivalent to:

$$\tilde{\tilde{\mathbf{M}}}\tilde{\tilde{\mathbf{z}}}^{\mathbf{P}_2} = \tilde{\tilde{\mathbf{r}}}^{\mathbf{P}_2} \tag{5.8}$$

where $\tilde{\tilde{\mathbf{M}}} = diag(\tilde{\tilde{\mathbf{T}}}_{1,1}, \tilde{\tilde{\mathbf{T}}}_{1,2} \ldots, \tilde{\tilde{\mathbf{T}}}_{1,m}, \tilde{\tilde{\mathbf{T}}}_{2,1}, \ldots, \tilde{\tilde{\mathbf{T}}}_{2,m}, \ldots, \tilde{\tilde{\mathbf{T}}}_{n,m})$, with $\tilde{\tilde{\mathbf{T}}}_{j,k},\ j = 1, \ldots, n,\ k = 1, \ldots, m$ being $l \times l$ tridiagonal matrices of the form:

$$\tilde{\tilde{\mathbf{T}}}_{j,k} = \begin{bmatrix} \tilde{\lambda}_{1,j,k} & -\delta_1 & & & \\ -\delta_1 & \tilde{\lambda}_{2,j,k} & -\delta_2 & & \\ & \cdot & \cdot & \cdot & \\ & & -\delta_{l-1} & \tilde{\lambda}_{l-1,j,k} & -\delta_l \\ & & & -\delta_l & \tilde{\lambda}_{l,j,k} \end{bmatrix} \tag{5.9}$$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Algorithm 8 Fast Transform algorithm for the preconditioner solve step $\mathbf{Mz} = \mathbf{r}$ where $\mathbf{M}$ is of the form (5.3)

1: Partition the RHS vector $\mathbf{r}$ into lm sub-vectors $\mathbf{r}_i$ of size n, and perform DCT-II transform ($\mathbf{Q}_n^T \mathbf{r}_i$) on each sub-vector to obtain transformed vector $\tilde{\mathbf{r}}$.
2: Partition vector $\tilde{\mathbf{r}}$ into l sub-vectors $\tilde{\mathbf{r}}_i$ of size mn, and permute each sub-vector by permutation $\mathbf{P}$, which orders elements as $1, n+1, \ldots, (m-1)n+1,\ 2, n+2, \ldots, (m-1)n+2,\ \ldots,\ n, n+n, \ldots, (m-1)n+n$, in order to obtain vector $\tilde{\mathbf{r}}^{\mathbf{P}_1}$.
3: Partition vector $\tilde{\mathbf{r}}^{\mathbf{P}_1}$ into ln sub-vectors $\tilde{\mathbf{r}}_i^{\mathbf{P}_1}$ of size m, and perform DCT-II transform ($\mathbf{Q}_m^T \tilde{\mathbf{r}}_i^{\mathbf{P}_1}$) on each sub-vector to obtain transformed vector $\tilde{\tilde{\mathbf{r}}}$.
4: Permute vector $\tilde{\tilde{\mathbf{r}}}$ by applying permutation $\mathbf{P}_2$, which orders elements as $1, mn+1, 2mn+1, \ldots, (l-1)mn+1,\ 2, mn+2, 2mn+2, \ldots, (l-1)mn+2,\ \ldots,\ mn, mn+mn, 2mn+mn, \ldots, (l-1)mn+mn$, in order to obtain vector $\tilde{\tilde{\mathbf{r}}}^{P_2}$.
5: Solve the mn tridiagonal systems (5.8) with known coefficient matrices (5.9), in order to obtain vector $\tilde{\tilde{\mathbf{z}}}^{P_2}$.
6: Apply inverse permutation $\mathbf{P}_2^T$ on vector $\tilde{\tilde{\mathbf{z}}}^{P_2}$ so as to obtain vector $\tilde{\tilde{\mathbf{z}}}$.
7: Partition vector $\tilde{\tilde{\mathbf{z}}}$ into ln sub-vectors $\tilde{\tilde{\mathbf{z}}}_i$ of size m, and perform IDCT-II transform ($\mathbf{Q}_m \tilde{\tilde{\mathbf{z}}}_i$) on each sub-vector to obtain vector $\tilde{\mathbf{z}}^{\mathbf{P}_1}$.
8: Partition vector $\tilde{\mathbf{z}}^{\mathbf{P}_1}$ into l sub-vectors $\tilde{\mathbf{z}}_i^{\mathbf{P}_1}$ of size mn, and apply inverse permutation $\mathbf{P}_1^T$ on each sub-vector to obtain vector $\tilde{\mathbf{z}}$.
9: Partition vector $\tilde{\mathbf{z}}$ into lm sub-vectors $\tilde{\mathbf{z}}_i$ of size n, and perform IDCT-II transform ($\mathbf{Q}_n \tilde{\mathbf{z}}_i$) on each sub-vector to obtain final solution vector $\mathbf{z}$.

and $\tilde{\tilde{\mathbf{z}}}^{\mathbf{P}_2} = \mathbf{P}_2 \tilde{\tilde{\mathbf{z}}}, \tilde{\tilde{\mathbf{r}}}^{\mathbf{P}_2} = \mathbf{P}_2 \tilde{\tilde{\mathbf{r}}}$. The equivalence of the system $\mathbf{Mz} = \mathbf{r}$, with $\mathbf{M}$ as in (5.3), to the system (5.8), gives a procedure for fast solution of $\mathbf{Mz} = \mathbf{r}$ which is described in Algorithm 8.

5.3 Proposed Approach for Electro-Thermal Analysis

Our proposed methodology combines the PCG iterative solution method with two highly-efficient preconditioning mechanisms, one for electrical and one for thermal power grid analysis. The 2D Fast Transform-based preconditioner described in Section 4.4.1 is applied for power grid electrical analysis. On the other hand, due to the structure of the thermal equivalent of the power grid, a novel preconditioning mechanism is required. Typically, to model the thermal profile of the power grid, a chip is considered as comprising n layers, where each layer contains metal

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

lines and inter-layer insulator. The topmost layer is covered by a thermal insulation layer and the heat generated in the power grid is conducted away by the substrate (usually attached to a heat sink). By modeling each layer as was described in Section 5.1, the thermal grid is equivalent to a highly regular resistive network, with resistive branches connecting nodes in the x, y, and z axis. To create a preconditioner that will approximate the grid matrix, we substitute each horizontal and vertical thermal conductance with its average value in the corresponding layer. Moreover, we substitute each thermal conductance $g_{i,i+1}$ connecting nodes in adjacent layers (z axis) with their average value between the two layers. Fig. 5-2 is an example of a thermal grid with $n = 3$, $m = 3$ nodes in the x and y axis respectively and $l = 3$ layers and corresponds to the physical model of the power grid depicted in Fig. 4-1(i). Using the depicted numbering, the matrix that corresponds to the aforementioned grid is the following block-tridiagonal matrix:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & -g_{1,2}\mathbf{I}_{mn} & \\ -g_{1,2}\mathbf{I}_{mn} & \mathbf{M}_2 & -g_{2,3}\mathbf{I}_{mn} \\ & -g_{2,3}\mathbf{I}_{mn} & \mathbf{M}_3 \end{bmatrix}$$

where

$$\mathbf{M}_i = \begin{bmatrix} \mathbf{T}_i + g_i^v\mathbf{I}_n & -g_i^v\mathbf{I}_n & \\ -g_i^v\mathbf{I}_n & \mathbf{T}_i + 2g_i^v\mathbf{I}_n & -g_i^v\mathbf{I}_n \\ & -g_i^v\mathbf{I}_n & \mathbf{T}_i + g_i^v\mathbf{I}_n \end{bmatrix}, \quad i = 1, 2, 3$$

$$\mathbf{T}_1 = \begin{bmatrix} g_1^h + g_{1,2} & -g_1^h & \\ -g_1^h & 2g_1^h + g_{1,2} & -g_1^h \\ & -g_1^h & g_2^h + g_{1,2} \end{bmatrix}$$

$$\mathbf{T}_2 = \begin{bmatrix} g_2^h + g_{1,2} + g_{2,3} & -g_2^h & \\ -g_2^h & 2g_2^h + g_{1,2} + g_{2,3} & -g_2^h \\ & -g_2^h & g_2^h + g_{1,2} + g_{2,3} \end{bmatrix}$$

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

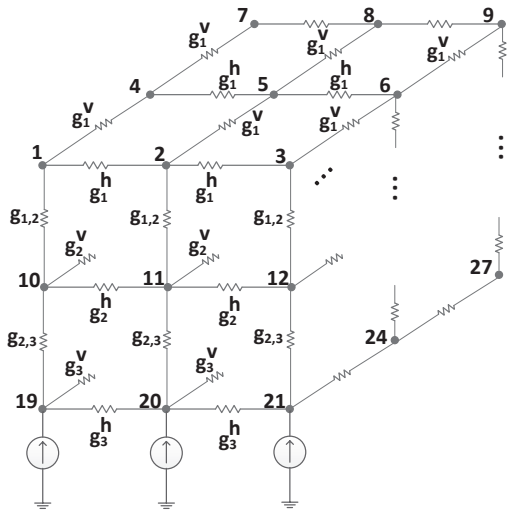


Figure 5-2: Example of the thermal equivalent of the 3D power grid from Fig. 4-1(i) that is used for preconditioning.

$$\mathbf{T}_3 = \begin{bmatrix} g_3^h + g_{2,3} & -g_3^h & \\ -g_3^h & 2g_3^h + g_{2,3} & -g_3^h \\ & -g_3^h & g_3^h + g_{2,3} \end{bmatrix}$$

Similarly to the power grid preconditioner, the form of the above matrix is identical to matrix (5.3). As a result, if such a matrix is used as the preconditioner $\mathbf{M}$ for the thermal analysis procedure, it can be efficiently solved through utilization of a Fast Transform solver, as was described in Section 5.2.

5.3.1 Procedure Implementation

Once the preconditioners for the power and the thermal grid have been created, the proposed approach executes the electro-thermal loop in Fig. 5-1. At each step this requires the solution of the power grid at step 1 and the solution of the thermal grid at step 3. An off-the-self implementation of the PCG method can be used for both steps, with an external call for the preconditioner-solve step $\mathbf{Mz} = \mathbf{r}$. The latter corresponds to Algorithm 6 for power grid electrical simulation and algorithm in Algorithm 8 for power grid thermal simulation.

As in the case of FTCG and FTCG-3D, the proposed preconditioning algorithm for electro-thermal offers a number of significant advantages. Owing the special

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

construction that allow utilization of Fast Transform solvers, solution of the preconditioned systems of the power and the thermal grid offer ample parallelism, both at data- and task-level. Both FFT and the tridiagonal system solution are highly-parallel algorithms that offer abundant level of data-level parallelism [15] [44]. Thus, the proposed method can efficiently utilize the computational resources found in massively parallel architectures, thus greatly accelerating the simulation process. This comes in contrast with most widely-used preconditioning methods, such as incomplete factorizations, which have limited parallelism.

As far as task-level parallelism is concerned, Algorithm 6 and Algorithm 8 entail a number of independent one-dimensional DCT-II and IDCT-II transforms as well as the solution of a large number of independent tridiagonal systems. This translates to additional task-level parallelism, which can result to further acceleration on multi-PCG systems for the preconditioner solve step as the independent transforms and tridiagonal solvers can be executed in parallel, requiring limited communication between different GPUs.

One other salient feature of the proposed preconditioners (apart from the near-optimal complexity of solving the systems $\mathbf{Mz} = \mathbf{r}$ and the parallelization opportunities) is that there is no need for explicit storage of the preconditioner matrix $\mathbf{M}$, which comes in contrast with most standard preconditioners. As it is easily observed, only the eigenvalues and the values γ_i and δ_i of $\mathbf{M}$ matrices in (4.6) and (5.3) respectively are necessary in the execution of Algorithm 6 and Algorithm 8. Thus, only limited storage is required for the preconditioners. A small memory footprint is very important for mapping the algorithm onto architectures with limited available memory space such as GPUs.

5.4 Experimental Evaluation

To evaluate the efficiency of the proposed methodology for combined electro-thermal simulation, we compared three methods for solving the linear systems for the power and the thermal grid (steps 1 and 3 in Fig. 5-1): the PCG method with

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

zero-fill Incomplete Cholesky preconditioner (ICCG), the proposed method of using PCG with the Fast Transform preconditioners (ET-FTCG), and CHOLMOD [12] which is a state-of-the-art direct solver for sparse SPD linear systems. Each method was ported on a GPU platform and the only part that is executed on the CPU is the construction of the power and thermal grid preconditioners for ET-FTCG and ICCG. Subsequently, the CPU is responsible for transferring the appropriate data to the GPU. We have used the CUDA library [2] (version 4.2, along with CUBLAS, CUSPARSE and CUFFT libraries) for mapping the ICCG and the ET-FTCG algorithm on the GPU.

Due to the lack of a set of available benchmarks for electro-thermal analysis, we have created a set of synthetic benchmarks, based on the modeling described in [27] (namely 10% of the wiring resources are used for the power grid), with size ranging from 3.1M to 20.9M-nodes. For the thermal grid, the length Δ_z of the grid rectangle was selected equal to the layer thickness (which can be variable), while the lengths Δ_x and Δ_y were chosen equal to the smallest routing width/pitch within a layer. We executed all experiments on a Linux workstation, comprising an Intel Core i7 processor running at 2.4GHz (6 cores and 24GB main memory) and an NVIDIA Tesla C2075 GPU with 5GB of main memory. Table 5.1 presents the results from the evaluation of the aforementioned methods on the set of benchmark circuits. The number of nodes in each circuit ($Nodes$) refers to the total number of nodes in the power grid, while execution time ($Time$) refers to the average time required for solution at each electro-thermal loop iteration, including any overhead for matrix factorization (in CHOLMOD) and preconditioner construction (in iterative methods).

As we can observe, CHOLMOD (the direct solution method) was able to simulate only the smaller benchmark circuit. Due to its excessive memory requirements, analysis of larger benchmarks was infeasible. On the other hand, both ICCG and ET-FTCG owing to the limited memory requirements were able to simulate the complete set of benchmarks. Moreover, they achieved a speed-up of $7X$ and $66.1X$ respectively.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Table 5.1: Runtime results for the three solvers for electro-thermal analysis. *Iter.* is the average number of iterations (total number of iterations in each iteration over the number of iterations required for convergence of the electro-thermal loop) required for convergence of each iterative method. *Time* denotes the average time required for the solution at each iteration, while Spd_{CHOL} and Spd_{ICCG} denote the speedup of ET-FTCG over CHOLMOD and ICCG respectively. The convergence tolerance for iterative solvers was 10^{-6} and convergence was achieved in all cases.

Benchmark		CHOLMOD	ICCG		ET-FTCG			
	Nodes	*Time* (*s*)	*Iter.*	*Time* (*s*)	*Iter.*	*Time* (*s*)	Spd_{CHOL}	Spd_{ICCG}
ckt1	3.1M	105.8	201	15.1	62	1.6	66.1X	9.4X
ckt2	6.3M	N/A	296	58.1	63	3.7	N/A	15.7X
ckt3	14.6M	N/A	465	214.1	62	10.6	N/A	20.1X
ckt4	16.7M	N/A	495	259.4	62	12.5	N/A	20.8X
ckt5	18.8M	N/A	536	314.2	62	14.6	N/A	21.5X
ckt6	19.9M	N/A	540	331.6	59	15.2	N/A	21.8X
ckt7	20.9M	N/A	551	359.5	61	16.2	N/A	22.2X

If we restrict our comparison to the iterative methods, we can observe that the proposed method was able to greatly reduce the number of iterations required for convergence. Compared with general purpose preconditioning methods such as Incomplete Cholesky factorization, the proposed preconditioners take into account the topology characteristics of the power and the thermal grid. As a result, they are able to approximate them faithfully enough and reduce the required number of iterations. Actually, the number of iterations remains constant (there is only a slight variation) as the size of the power delivery network increases, which is a testament to the efficiency of the proposed preconditioning mechanism.

Moreover, owing to their inherent parallelism, the proposed preconditioners can utilize the vast amount of computational resources found in massively parallel architectures, such as GPUs. Thus, their efficacy is increased with the increasing circuit size. ET-FTCG was able to achieve a speed-up ranging between $9.4X$ and $22.2X$ over ICCG for our benchmark circuits. On the contrary, ICCG was not able to fully utilize the GPU resources due to the limited parallelism found in the triangular solution algorithm.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

This dissertation presented new methodologies for electrical and electro-thermal analysis of large-scale power delivery networks found in contemporary ICs. Its contributions are the following three highly parallel and efficient algorithms for power grid analysis:

- FTCG and FTCG-3D for electrical analysis of power delivery networks with negligible and significant via resistances respectively. The proposed algorithms combine an iterative linear system solution method with two problem-specific preconditioners that take into account the geometry structure of the power grid in order to accelerate the convergence of the iterative method. In addition, owing to their special structure that is based on a Fast Transform solver, they entail a large degree of multi-level parallelism. As a result, mapping onto parallel architectures is very efficient and can provide additional acceleration. FTCG and FTCG-3D are able to achieve up to three orders of magnitude acceleration in simulation time when compared to CHOLMOD (the state-of-the-art direct solver) and up to two orders of magnitude compared with the PCG method with either an Incomplete Cholesky or a support graph-based preconditioner.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

- ET-FTCG for combined electro-thermal analysis of the power delivery network. ET-FTCG combines the FTCG method with a novel preconditioning algorithm for thermal analysis of the power grid. Again, we utilize a Fast Transform-based preconditioner that allows for acceleration of the convergence rate and high parallel efficiency. As a result, the proposed algorithm is able to simulate power delivery networks with up to 22M nodes and achieve an execution time that is one order of magnitude lower than the time required for analysis with a state-of-the-art algorithm.

6.2 Future Directions

In the future, we plan to extend the research presented in this dissertation towards the following directions:

- 3D ICs: Three-dimensional chip design emerges as a promising technology, able to reduce interconnect issues found in modern ICs. In this technology, Through Silicon Vias (TSVs) are used in order to connect several layers, thus reducing interconnection length between vertically stacked integrated circuits. As a result, TSV-based 3D ICs appear as a better alternative for the design of high-speed IC systems with smaller form factors and enhanced performance. The power delivery network in such circuits form a vertical stack as it comprises a number of local power delivery networks, with TSVs providing the connections between adjacent layers. Essentially, this corresponds to a 3D grid, with some irregularities due to missing connections. We plan to extend FTCG-3D in order to provide a highly efficient yet accurate GPU-accelerated power delivery network simulation methodology with promising reduction in simulation time for power grid analysis of large-scale 3D TSV-based ICs.

- Full-chip thermal analysis: Due to the ever increasing chip complexity and the subsequent power density, contemporary ICs experience big temperature variations across the chip. Full-chip thermal analysis is an indispensable part

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

for the analysis of an IC as it provides useful insights for revealing hotspots, selecting the appropriate cooling technology, and avoiding excessive temperature variations. Furthermore, full-chip thermal simulation results can be used for more accurate power, timing and electromigration simulations. However, full-chip thermal analysis down to the device and interconnect levels is very computational expensive as it requires the solution of a 3D dense and fine-grained mesh structure (comprising up to hundreds of millions nodes), which has excessive computational and memory requirements. Based on ET-FTCG, we plan to develop efficient GPU-accelerated algorithms that will be able to handle the heterogeneous multi-layer full-chip structure and will enable efficient full-chip thermal analysis on parallel architectures.

- Mapping onto heterogeneous systems: Owing to the diverse nature of parallel computing architectures (such as multi-core processors, GPUs, or even Field-Programmable Gate Arrayss (FPGAs)) it is very appealing to have heterogeneous systems that comprise dissimilar processors, each one incorporating its own parallel capabilities. Each one of these processors will be programmed with its own programming model and hybrid programming languages and libraries will be required in order to orchestrate execution of an algorithm on such architectures. We plan to investigate efficient ways for mapping the proposed algorithms on heterogeneous architectures using a variety of programming models and languages. The proposed algorithms entail a large degree of both data- and task-level parallelism and we anticipate that mapping onto heterogeneous architectures will provide additional acceleration, thus making analysis of even larger power delivery networks extremely fast.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Mathematical Proofs

In this section we provide a detailed proof regarding the eigenvalues and the eigenvectors of the matrix from (4.7), which are given in (4.8) and (4.9) respectively. If $\mathbf{A}$ is a $n \times n$ square matrix with eigenvalues $\lambda_j, j = 1, \ldots, n$ (not necessarily distinct) and eigenvectors $\mathbf{q}_j$, $j = 1, \ldots, n$, then for the matrix $\mathbf{B} = \alpha\mathbf{A} + \beta\mathbf{I}$ it holds:

$$\mathbf{B}\mathbf{q}_j = (\alpha\mathbf{A} + \beta\mathbf{I})\mathbf{q}_j = \alpha\lambda_j\mathbf{q}_j + \beta\mathbf{q}_j = (\alpha\lambda_j + \beta)\mathbf{q}_j$$

which means that the eigenvalues of $\mathbf{B}$ are equal to $\alpha\lambda_j + \beta$, $j = 1, \ldots, n$, and the corresponding eigenvectors are $\mathbf{q}_j$, $j = 1, \ldots, n$. Thus, in order to prove (4.8) and (4.9), it is sufficient to show that for the $n \times n$ tridiagonal matrix:

$$\mathbf{T} = \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \cdot & \cdot & \cdot & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix}$$

the eigenvalues are equal to:

$$\lambda_j = 2 - 2\cos\frac{(j-1)\pi}{n},\ j = 1, \ldots, n \tag{6.1}$$

and the eigenvectors are given by (4.9).

By definition, if λ is an eigenvalue and $\mathbf{q}$ is an eigenvector of $\mathbf{T}$ then it holds:

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$$\mathbf{Tq} = \lambda\mathbf{q}$$

or:

$$\begin{cases} q_1 - q_2 = \lambda q_1 \\ -q_{k-1} + 2q_k - q_{k+1}, \ k = 2, \ldots, n-1 \\ -q_{n-1} + q_n = \lambda q_n \end{cases}$$

which can be written as:

$$q_{k+1} + (\lambda - 2)q_k + q_{k+1} = 0, \ k = 1, \ldots, n \tag{6.2}$$

where $q_0 = q_1$ and $q_{n+1} = q_n$.

The latter is a sequence for q_k, $k = 1, \ldots, n$ which is defined recursively. This recursive sequence or recurrence is effectively a second-order linear difference equation [6] with characteristic equation:

$$\rho^2 + (\lambda - 2)\rho + 1 = 0 \tag{6.3}$$

Let ρ_1, ρ_2 denote the roots of (6.3), which are assumed to be distinct ($\rho_1 \neq \rho_2$), i.e.:

$$\rho_{1,2} = \frac{2 - \lambda \pm \sqrt{(\lambda - 2)^2 - 4}}{2} \tag{6.4}$$

with $\lambda \neq 0$ and $\lambda \neq 4$ [we will consider the cases $\lambda = 0$ ($\Rightarrow \rho_1 = \rho_2 = 1$) and $\lambda = 0$ ($\Rightarrow \rho_1 = \rho_2 = -1$) in the end]. An explicit formula for q_k can be found by the general solution of (6.3):

$$q_k = c_1\rho_1^k + c_2\rho_2^k, \ k = 1, \ldots, n \tag{6.5}$$

where c_1, c_2 are constants (to be determined by the end conditions). Now, the eigenvectors of a matrix are always defined up to a multiplicative constant [since if $\mathbf{q}$ is an eigenvector corresponding to eigenvalue λ then $c\mathbf{q}$ ($\forall c \neq 0$) is also an eigenvector corresponding to λ]. Thus, in order to determine c_1, c_2 and calculate

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

the eigenvectors $\mathbf{q}$ from (6.4), we can arbitrarily pick their first component to be $q_1 = 1$ and additionally use the left-end condition $q_0 = q_1 = 1$. From (6.5) we have for $k = 0$ and $k = 1$:

$c_1 + c_2 = 1$ and $c_1\rho_1 + c_2\rho_2 = 1$

which by solving w.r.t. c_1, c_2 (and considering that $\rho_2 - \rho_1 \neq 0$) gives:

$c_1 = \frac{\rho_2 - 1}{\rho_2 - \rho_1}$ and $c_2 = \frac{1 - \rho_1}{\rho_2 - \rho_1}$

and therefore:

$$q_k = \frac{\rho_2 - 1}{\rho_2 - \rho_1}\rho_1^k + \frac{1 - \rho_1}{\rho_2 - \rho_1}\rho_2^k,\ k = 1, \ldots, n \tag{6.6}$$

The right-end condition $q_{n+1} = q_n$ can be used to determine the eigenvalues $\lambda_j,\ j = 1, \ldots, n$, since the recurrence (6.2) is specified up to parameter λ. From (6.6) we have:

$$\begin{aligned} q_{n+1} = q_n &\Rightarrow \frac{\rho_2 - 1}{\rho_2 - \rho_1}\rho_1^n\rho_1 + \frac{1 - \rho_1}{\rho_2 - \rho_1}\rho_2^n\rho_2 = \frac{\rho_2 - 1}{\rho_2 - \rho_1}\rho_1^n + \frac{1 - \rho_1}{\rho_2 - \rho_1}\rho_2^n \\ &\Rightarrow \frac{\rho_2 - 1}{\rho_2 - \rho_1}\rho_1^n(\rho_1 - 1) + \frac{\rho_1 - 1}{\rho_2 - \rho_1}\rho_2^n(\rho_2 - 1) \end{aligned}$$

and since $\rho_1 \neq \rho_2$ and $\rho_2 \neq 1$, $\rho_2 \neq 1$, it is:

$$(\frac{\rho_1}{\rho_2})^n = 1 \tag{6.7}$$

which means that the ratio $\frac{\rho_1}{\rho_2}$ is equal to the n complex roots of unity, without the root $\frac{\rho_1}{\rho_2} = 1$ (which gives $\rho_1 = \rho_2$), i.e. $\frac{\rho_1}{\rho_2} = \exp(i\frac{2j\pi}{n}) = \cos\frac{2j\pi}{n} + i\sin\frac{2j\pi}{n}$, $j = 1, \ldots, n-1$ [i is the imaginary unit ($i^2 = -1$]. Instead, now, of using the actual expressions of ρ_1, ρ_2 from (6.4) into (6.7), it is easier to use their equivalent sum and product from the quadratic equation (6.3), i.e.

$$\rho_1 + \rho_2 = 2 - \lambda \tag{6.8i}$$

$$\rho_1\rho_2 = 1 \tag{6.8ii}$$

From (6.7) and (6.8ii) we have:

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$$(\frac{\rho_1}{\rho_2})^n = 1 \Rightarrow (\rho_1)^{2n} = 1 \Rightarrow \rho_1 = \exp{(i\frac{j\pi}{n})},\ j = 1, \ldots, n-1$$

or:

$$\rho_1 = \exp{(i\frac{(j-1)\pi}{n})},\ j = 2, \ldots, n \tag{6.9i}$$

$$\rho_2 = \frac{1}{\rho_1} = \exp{(-i\frac{(j-1)\pi}{n})},\ j = 2, \ldots, n \tag{6.9ii}$$

and from the latter and (6.8i):

$$\begin{aligned} &\exp{(i\frac{(j-1)\pi}{n})} + \exp{(-i\frac{(j-1)\pi}{n})} = 2 - \lambda \\ &\Rightarrow 2\cos\frac{(j-1)\pi}{n} = 2 - \lambda,\ j = 2, \ldots, n \end{aligned}$$

from which we obtain (6.1) (without the case $j = 1$).

Also, by substituting (6.9i) and (6.9ii) into (6.6), we have:

$$\begin{aligned} q_k &= \frac{[\exp{(-i(j-1)\pi/n)} - 1]\exp{(ik(j-1)\pi/n)}}{\exp{(-i(j-1)\pi/n)} - \exp{(i(j-1)\pi/n)}} \\ &+ \frac{[1 - \exp{(i(j-1)\pi/n)}]\exp{(-ik(j-1)\pi/n)}}{\exp{(-i(j-1)\pi/n)} - \exp{(i(j-1)\pi/n)}} \\ &= \frac{2i\sin{((k-1)(j-1)\pi/n)} - 2i\sin{(k(j-1)\pi/n)}}{-2i\sin{((j-1)\pi/n)}} \\ &= \frac{2\cos{((2k-1)(j-1)\frac{\pi}{2n})}\sin{((j-1)\frac{\pi}{2n})}}{2\sin{((j-1)\frac{\pi}{2n})}\cos{((j-1)\frac{\pi}{2n})}} \end{aligned}$$

or:

$$q_k = [\frac{1}{\cos{((j-1)\frac{\pi}{2n})}}]\cos\frac{(2k-1)(j-1)\pi}{2n},\ k = 1, \ldots, n \tag{6.10}$$

The above gives the k-th component of the j-th eigenvector $\mathbf{q}_j$ of $\mathbf{T}$ for which $q_1 = 1$ (according to our initial arbitrary choice). In order to get the normalized eigenvector (with unit length $\|\mathbf{q}_j\|_2 = 1$), we notice that:

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$$\|\mathbf{q}_j\|_2 = \sqrt{\frac{1}{\cos^2\left((j-1)\frac{\pi}{2n}\right)}\sum_{k=1}^{n}\cos^2\frac{(2k-1)(j-1)\pi}{2n}}$$
$$= \frac{1}{\cos\left((j-1)\frac{\pi}{2n}\right)}\sqrt{\frac{n}{2}}$$

and thus by multiplying (6.10) by $\sqrt{\frac{2}{n}}\cos\left((j-1)\frac{\pi}{2n}\right)$, we obtain (4.9) (without the case $j=1$).

The case of identical roots $\rho_1 = \rho_2 \equiv \rho$ of (6.3) must be treated separately, since then the general solution of the recurrence of (6.2) is:

$$q_k = c_1\rho^k + c_2 k\rho^k,\ k = 1, \ldots, n \tag{6.11}$$

As mentioned previously, we have two possibilities for identical roots:

- $\lambda = 0$: In this case, $\rho_1 = \rho_2 \equiv \rho = 1$ and the solution of (6.11) becomes $q_k = c_1 + c_2 k,\ k = 1, \ldots, n$. By choosing $q_1 = 1$ and using the left-end condition $q_0 = q_1 = 1$, we have for $k = 0$ and $k = 1$:

$$\left.\begin{aligned} c_1 &= 1 \\ c_1 + c_2 &= 1 \end{aligned}\right\} \Rightarrow \begin{aligned} c_1 &= 1 \\ c_2 &= 0 \end{aligned}$$

  and thus $q_k = 1,\ k = 1, \ldots, n$, i.e. the vector $\mathbf{q} = [1, \ldots, 1]^T$. This solution also satisfies the right-end condition $q_{n+1} = q_n$, which means that the pair $\lambda = 0$ and $\mathbf{q} = [1, \ldots, 1]^T$ is a valid eigenvalue-eigenvector pair for the matrix $\mathbf{T}$ (the fact that $\lambda = 0$ actually means that $\mathbf{T}$ is singular). This pair can be obtained from (6.1) and (4.9) for $j = 1$ (the vector $[1, \ldots, 1]^T$ is normalized by $\sqrt{\frac{1}{n}}$ since $\sqrt{\sum_{k=1}^{n} 1} = \sqrt{n}$), as is easily observed.

- $\lambda = 4$: In this case, $\rho_1 = \rho_2 \equiv \rho = -1$ and the solution of (6.11) becomes $q_k = c_1(-1)^k + c_2 k,\ k = 1, \ldots, n$. By choosing $q_1 = 1$ and using the left-end condition $q_0 = q_1 = 1$, we have for $k = 0$ and $k = 1$:

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

$$\left.\begin{aligned} c_1 &= 1 \\ -c_1 - c_2 &= 1 \end{aligned}\right\} \Rightarrow \begin{aligned} c_1 &= 1 \\ c_2 &= -2 \end{aligned}$$

and thus $q_k = (-1)^k - 2k(-1)^k,\ k = 1, \ldots, n$. This solution does not satisfy the right-end condition $q_{n+1} = q_n$, which means that the pair $\lambda = 4$ and $\mathbf{q}$ with q_k as above is not a valid eigenvalue-eigenvector pair for the matrix $\mathbf{T}$.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Publications

The research conducted for this Ph.D. dissertation resulted to the following patent application and publications:

Pattent:

- Konstantis Daloukas, Nestoras Evmorfopoulos, Panagiota Tsompanopoulou, and Georgios Stamoulis. "Large-scale power grid analysis on parallel architectures", October 17 2013. US Patent App. 14/056,667

Journal Publications:

- Konstantis Daloukas, Nestor Evmorfopoulos, Panagiota Tsompanopoulou, and George I. Stamoulis. "Fast Transform-Based Preconditioning Approaches for Large-Scale Power Grid Analysis on Massively Parallel Architectures". IEEE Transactions on COMPUTER-AIDED DESIGN of Integrated Circuits and Systems (TCAD). (Under review)

Peer-Reviewed Conference Publications:

- Konstantis Daloukas, Nestor Evmorfopoulos, Panagiota Tsompanopoulou, and George I. Stamoulis. "A 3-D Fast Transform-Based Preconditioner for Large-Scale Power Grid Analysis on Massively Parallel Architectures". International Symposium on Quality Electronic Design (ISQED). Santa Clara, CA, March 2014

- Konstantis Daloukas, Alexia Marnari, Nestor Evmorfopoulos, Panagiota Tsompanopoulou, and George I. Stamoulis. "A Parallel Fast Transform-Based Preconditioning Approach for Electrical-Thermal Co-Simulation of Power Delivery

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

Networks". Proceedings of Design Automation and Test In Europe Conference (DATE). Grenoble, France, March 2013.

- Konstantis Daloukas, Nestor Evmorfopoulos, George Drasidis, Michalis Tsiampas, Panagiota Tsompanopoulou, and George I. Stamoulis. "Fast Transform-Based Preconditioners for Large-Scale Power Grid Analysis on Massively Parallel Architectures". Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD). San Jose, CA, U.S.A., November 2012. (**Best Paper Award Nominee**)

Posters:

- Konstantis Daloukas, Nestoras Evmorfopoulos, Panagiota Tsompanopoulou, George Stamoulis. "Fast Transform-Based Solvers as Parallel Preconditioners for Large-Scale Power Grid Analysis on Massively Parallel Architectures". Poster presented in the ACM Student Research Competition, held in conjuction with the 49th International Design Automation Conference (DAC), San Francisco, CA, U.S.A., June 2012.

Also, our research led to the following publication that is not related with the content of this dissertation:

- Ifigeneia Apostolopoulou, Konstantis Daloukas, Nestor Evmorfopoulos, and George I. Stamoulis. "Selective Inversion of Inductance Matrix for Large-Scale Sparse RLC Simulation". Design Automation Conference (DAC 2014). San Francisco, CA, U.S.A., June 2014.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

References

[1] Intel Math Kernel Library.

[2] NVIDIA CUDA Programming Guide, CUSPARSE, CUBLAS, and CUFFT Library User Guides.

[3] Marco Ament, Gunter Knittel, Daniel Weiskopf, and Wolfgang Strasser. A Parallel Preconditioned Conjugate Gradient Solver for the Poisson Problem on a Multi-GPU Platform. In Euromicro Conference on Parallel, Distributed and Network-based Processing, 2010.

[4] O. Axelsson and A. Barker. Finite Element Solution of Boundary Value Problems. Theory and Computation. Academic Press, 1984.

[5] K. Banerjee and A. Mehrotra. Global (interconnect) warming. Circuits and Devices Magazine, IEEE, 17(5):16–32, 2001.

[6] Paul Mason Batchelder. An Introduction to Linear Difference Equations. New York, 1967.

[7] Michele Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. Journal of Computational Physics, 182(2):418–477, 2002.

[8] Michele Benzi, Carl D. Meyer, and Miroslav Tuma. A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method. SIAM Journal on Scientific Computing, 17(5):1135–1149, 1996.

[9] John C. Butcher. Numerical Methods for Ordinary Differential Equations. Wiley, 2008.

[10] Raymond H. Chan and C. K. Wong. Sine Transform Based Preconditioners for Elliptic Problems. Numerical Linear Algebra with Applications, 4(5):351–368, 1997.

[11] Tsung-Hao Chen and Charlie Chung-Ping Chen. Efficient Large-Scale Power Grid Analysis Based on Preconditioned Krylov-Subspace Iterative Methods. In ACM/IEEE Design Automation Conf., 2001.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

[12] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. ACM Transactions on Mathematical Software (TOMS), 35(3):22:1–22:14, 2008.

[13] Chung-Han Chou, Nien-Yu Tsai, Hao Yu, Che-Rung Lee, Yiyu Shi, and Shih-Chieh Chang. On the Preconditioner of Conjugate Gradient Method - A Power Grid Simulation Perspective. In Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on, 2011.

[14] Christina C. Christara and Kit Sun Ng. Fast Fourier Transform Solvers and Preconditioners for Quadratic Spline Collocation. BIT Numerical Mathematics, 42(4):702–739, 2002.

[15] Zhong Cui-xiang, Han Guo-qiang, and Huang Ming-he. Some New Parallel Fast Fourier Transform Algorithms. In Int. Conf. on Parallel and Distributed Computing, Applications and Technologies, 2005.

[16] Zhuo Feng and Zhiyu Zeng. Parallel Multigrid Preconditioning on Graphics Processing Units (GPUs) for Robust Power Grid Analysis. In ACM/IEEE Design Automation Conf., 2010.

[17] Zhuo Feng, Zhiyu Zeng, and Peng Li. Parallel On-Chip Power Distribution Network Analysis on Multi-Core-Multi-GPU Platforms. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 19(10):1823–1836, 2011.

[18] Nico Galoppo, Naga K. Govindaraju, Michael Henson, and Dinesh Manocha. LU-GPU: Efficient Algorithms for Solving Dense Linear Systems on Graphics Hardware. In Proceedings of the 2005 ACM/IEEE conference on Supercomputing, IEEE Computer Society, page 3, 2005.

[19] Xiang Hu, Wenbo Zhao, Peng Du, A. Shayan, and Chung-Kuan Cheng. An Adaptive Parallel Flow for Power Distribution Network Simulation Using Discrete Fourier Transform. In Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific, 2010.

[20] J.N. Kozhaya, S.R. Nassif, and F.N. Najm. A Multigrid-Like Technique for Power Grid Analysis. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 21(10):1148–1160, 2002.

[21] M. Krotkiewski and M. Dabrowski. Efficient Solution of Poisson's Equation on Modern GPUs Using Structured Grids.

[22] Duo Li, Sheldon X-D Tan, and Bruce McGaughy. ETBR: Extended Truncated Balanced Realization Method for On-Chip Power Grid Network Analysis. In Proceedings of the conference on Design, automation and test in Europe, 2008.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

[23] R. Li and Y. Saad. GPU-Accelerated Preconditioned Iterative Linear Solvers. Technical report, Minnesota Supercomputer Institute, University of Minnesota, 2010.

[24] Xue-Xin Liu, Zao Liu, S.X.-D. Tan, and J. Gordon. Full-chip Thermal Analysis of 3D ICs with Liquid Cooling by GPU-accelerated GMRES Method. In Quality Electronic Design (ISQED), 2012.

[25] Xue-Xin Liu, Hai Wang, and Sheldon X.-D. Tan. Parallel Power Grid Analysis Using Preconditioned GMRES Solver on CPU-GPU Platforms. In Computer-Aided Design (ICCAD), 2013 IEEE/ACM International Conference on, 2013.

[26] S. R. Nassif and J. N. Kozhaya. Fast Power Grid Simulation. In Proceedings of the 37th Annual Design Automation Conference, 2000.

[27] S.R. Nassif. Power Grid Analysis Benchmarks. In Asia and South Pacific Design Automation Conf., 2008.

[28] Haifeng Qian, Sani R. Nassif, and Sachin S. Sapatnekar. Power Grid Analysis Using Random Walks. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 24(8), 2005.

[29] Haifeng Qian and S.S. Sapatnekar. Fast Poisson Solvers for Thermal Analysis. In Computer-Aided Design (ICCAD), 2010 IEEE/ACM International Conference on, 2010.

[30] Pablo Quesada-Barriuso, Julián Lamas-Rodríguez, Dora B. Heras, Montserrat Bóo1, and Francisco Argüello. Selecting the Best Tridiagonal System Solver Projected on Multi-Core CPU and GPU Platforms. In Int. Conf. on Parallel and Distributed Processing Techniques and Applications (as part of WorldComp'2011 Conference), 2011.

[31] Yousef Saad. Iterative Methods for Sparse Linear Systems. SIAM, 2003.

[32] Jin Shi, Yici Cai, Wenting Hou, Liwei Ma, Sheldon X.-D. Tan, Pei-Hsin Ho, and Xiaoyi Wang. GPU friendly Fast Poisson Solver for Structured Power Grid Network Analysis. In ACM/IEEE Design Automation Conf., 2009.

[33] Jin Shi, Yici Cai, Sheldon X.-D. Tan, Jeffrey Fan, and Xianlong Hong. Pattern-Based Iterative Method for Extreme Large Power/Ground Analysis. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 26(4):680–692, 2007.

[34] Kai Sun, Quming Zhou, Kartik Mohanram, and Danny C. Sorensen. Parallel Domain Decomposition for Simulation of Large-Scale Power Grids. In ACM/IEEE Design Automation Conf., 2007.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

[35] Llewellyn Hilleth Thomas. Elliptic Problems in Linear Difference Equations Over a Network. Watson Sci. Comput. Lab. Rept., Columbia University, New York, 1949.

[36] Charles Van Loan. Computational Frameworks for the Fast Fourier Transform. SIAM, 1992.

[37] Janet M. Wang and Tuyen V. Nguyen. Extended Krylov Subspace Method for Reduced Order Analysis of Linear Circuits with Multiple Sources. In Proceedings of the 37th Annual Design Automation Conference, 2000.

[38] Jia Wang. Deterministic Random Walk Preconditioning for Power Grid Analysis. In Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on, 2012.

[39] Jianyong Xie and M. Swaminathan. Electrical-Thermal Co-Simulation of 3D Integrated Systems With Micro-Fluidic Cooling and Joule Heating Effects. Components, Packaging and Manufacturing Technology, IEEE Transactions on, 1(2):234–246, 2011.

[40] Xuanxing Xiong and Jia Wang. Parallel Forward and Back Substitution for Efficient Power Grid Simulation. In Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on, 2012.

[41] Jianlei Yang, Yici Cai, Qiang Zhou, and Jin Shi. Fast Poisson Solver Preconditioned Method for Robust Power Grid Analysis. In Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on, 2011.

[42] Jianlei Yang, Zuowei Li, Yici Cai, and Qiang Zhou. PowerRush: A Linear Simulator for Power Grid. In Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on, 2011.

[43] Ting Yu, Zigang Xiao, and M.D.F. Wong. Efficient parallel power grid analysis via Additive Schwarz Method. In Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on, 2012.

[44] Yao Zhang, Jonathan Cohen, and John D. Owens. Fast Tridiagonal Solvers on the GPU. In ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2010.

[45] Min Zhao, Rajendran V. Panda, Sachin S. Sapatnekar, and David Blaauw. Hierarchical Analysis of Power Distribution Networks. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 21(2), 2002.

[46] Xueqian Zhao, Jia Wang, Zhuo Feng, and Shiyan Hu. Power Grid Analysis with Hierarchical Support Graphs. In Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on, 2011.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7

[47] Yu Zhong and Martin D. F. Wong. Thermal-Aware IR Drop Analysis in Large Power Grid. In ISQED'08, 2008.

[48] Cheng Zhuo, Jiang Hu, Min Zhao, and Kangsheng Chen. Power Grid Analysis and Optimization Using Algebraic Multigrid. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 27(4):738–751, 2008.

[49] Dan Zou, Yong Dou, Song Guo, Rongchun Li, and Lin Deng. Supernodal Sparse Cholesky Factorization on Graphics Processing Units. Concurrency and Computation: Practice and Experience, 2013.

Institutional Repository - Library & Information Centre - University of Thessaly
09/12/2017 11:34:19 EET - 137.108.70.7