Fast Transform-Based Preconditioners for Large-Scale Power Grid Analysis on Massively Parallel Architectures

Konstantis Daloukas
Nestor Evmorfopoulos
George Drasidis
Michalis Tsiampas
Panagiota Tsompanopoulou
George George
Stamoulis

Department of Computer and Communications Engineering
University of Thessaly
Volos, Greece
Introduction

- Power grid analysis and verification are of outmost importance to ensure circuit proper functionality
- IR drop increases with technology and voltage scaling
  - Affects circuit reliability
- Static (DC) or transient simulation:
  - Compute the response of a circuit to a constant or time-varying stimulus
Motivation

- Static and dynamic simulation are very challenging for contemporary nano-scale ICs
  - Power grids comprise a very large number of nodes
  - Computationally-intensive process
- **Solution**: Harness the computational capabilities of parallel architectures to accelerate simulation
Motivation

• State-of-the-art simulators (e.g. SPICE) employ direct methods for power grid analysis
• Direct methods have limited parallelism and excessive memory requirements
• Iterative methods present a better alternative
  – Highly-parallel operations
  – Limited memory requirements
• However, their unpredictable convergence rate hinders their adoption
Contribution

• Present the design and development of \textbf{FTCG}, an efficient and highly-parallel algorithm for power grid analysis

• Combination of an iterative solution method with a Fast Transform-based preconditioner

• Advantages:
  – Accelerates convergence rate
  – Suitable for mapping onto parallel architectures
  – Limited memory requirements
Outline

• Introduction
• Theoretical Background
• Proposed Methodology
• Experimental Evaluation
• Conclusions
Power Grid Analysis

- Power delivery networks are modeled as 3D RC grids

- Modified Nodal Analysis (MNA):
  - $Gv = e$, DC Analysis
  - $Gv(t) + C \frac{dv(t)}{dt} = e(t)$, Transient Analysis
  - Symmetric Positive Definite (SPD) linear systems that comprise a very large number of unknowns

22/11/2012

ICCAD 2012
Iterative Solution Methods

• Iterative methods constitute an attractive alternative for solving linear systems
  – Due to their limited computational and memory requirements compared to direct methods

• Conjugate Gradients (CG) method
  – For SPD systems, such as the power grid MNA system

• Comprises only inner and matrix-vector products

• Main disadvantage: Unpredictable convergence rate
Iterative Solution Methods

Preconditioning

• Preconditioning is used to accelerate convergence of iterative methods – Preconditioned CG (PCG)
  – Preconditioner matrix \( M \)
• Essentially solves the system \( M^{-1}Ax = M^{-1}b \)
• Preconditioner solve step \( Mz = r \) at each iteration receives the whole computational burden
• \( M \) matrix properties:
  – \( M \) provides a good approximation to \( A \)
  – \( Mz = r \) can be solved very efficiently
PCG Algorithm

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

- Efficient direct solution methods for the solution of the linear system $Mz = r$
- $M$ must employ a special form

$$M = \begin{bmatrix}
T_1 & \gamma_1 I & & \\
\gamma_1 I & T_2 & \gamma_2 I & \\
& \ddots & \ddots & \\
\gamma_{m-2} I & T_{m-1} & \gamma_{m-1} I & \\
\gamma_{m-1} I & T_m & & 
\end{bmatrix}, \quad T_i = \alpha_i \begin{bmatrix}
1 & -1 & & \\
-1 & 2 & -1 & \\
& \ddots & \ddots & \\
& -1 & 2 & -1 \\
& & & -1 & 1
\end{bmatrix} + \beta_i I$$

- Take advantage of the fact that the eigen-decomposition ($T_i = QA_iQ^T$) of $T_i$ is known beforehand
Fast Transform Solvers

- Multiply with $\text{diag}(Q^T, \ldots, Q^T)$

\[
\begin{bmatrix}
Q^T \\
\vdots \\
Q^T
\end{bmatrix}
M
\begin{bmatrix}
Q \\
\vdots \\
Q
\end{bmatrix}
\begin{bmatrix}
Q^T \\
\vdots \\
Q^T
\end{bmatrix}
\begin{bmatrix}
Q^T \\
\vdots \\
Q^T
\end{bmatrix}
= \begin{bmatrix}
Q^T \\
\vdots \\
Q^T
\end{bmatrix}
\begin{bmatrix}
Q^T \\
\vdots \\
Q^T
\end{bmatrix}
\begin{bmatrix}
Q^T \\
\vdots \\
Q^T
\end{bmatrix}
\]

\[
Q^T T_i Q = \Lambda_i
\]

\[
\begin{bmatrix}
\Lambda_1 & \gamma_1 I \\
\gamma_1 I & \Lambda_2 \\
\vdots & \ddots \\
\gamma_{m-2} I & \Lambda_{m-1} & \gamma_{m-1} I \\
\gamma_{m-1} I & \gamma_{m-1} I & \Lambda_m
\end{bmatrix}
\begin{bmatrix}
\tilde{z}
\end{bmatrix}
= \begin{bmatrix}
\tilde{r}
\end{bmatrix}
= \begin{bmatrix}
Q^T \\
\vdots \\
Q^T
\end{bmatrix}
\begin{bmatrix}
\tilde{z}
\end{bmatrix}
= \begin{bmatrix}
\tilde{r}
\end{bmatrix}
= \begin{bmatrix}
Q^T \\
\vdots \\
Q^T
\end{bmatrix}
\begin{bmatrix}
z
\end{bmatrix}
\begin{bmatrix}
r
\end{bmatrix}
\]

22/11/2012
Fast Transform Solvers

• Apply permutation matrix $P$ and create decoupled tridiagonal systems

• $P$: Permutation matrix that reorders the elements of a vector or the rows of a matrix as $1, n + 1, \ldots, (m - 1)n + 1, \ldots, n, n + n, (m - 1)n + n$
Fast Transform Solvers

\[ P \begin{bmatrix}
\Lambda_1 & \gamma_1 I \\
\gamma_1 I & \Lambda_2 \\
\vdots & \vdots \\
\gamma_{m-2} I & \Lambda_{m-1} \\
\gamma_{m-1} I & \Lambda_m \\
\end{bmatrix} P^T P \tilde{z} = P \tilde{r} \]

\[ \begin{bmatrix}
\tilde{T}_1 \\
\tilde{T}_2 \\
\vdots \\
\tilde{T}_n \\
\end{bmatrix} \]

\[ \tilde{z}^P = \tilde{r}^P \]
Fast Transform Solvers

- Equivalent Linear System

Represents decoupled tridiagonal systems where the coefficients (the eigenvalues $\lambda_{i,j}$) are known beforehand.
Fast Transform Solvers

• Where is the Fast Transform?

• Q: Eigenvector matrix

\[ Q = [q_1, \ldots, q_n] \]

\[ q_{j,k} = \begin{cases} \sqrt{\frac{1}{n}} \cos \left( \frac{(2k-1)(j-1)\pi}{2n} \right) \\ \sqrt{\frac{2}{n}} \cos \left( \frac{(2k-1)(j-1)\pi}{2n} \right) \end{cases} \]

\[ r = \begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix}, \quad z = \begin{bmatrix} z_1 \\ \vdots \\ z_m \end{bmatrix}, \quad \tilde{r} = \begin{bmatrix} \tilde{r}_1 \\ \vdots \\ \tilde{r}_m \end{bmatrix}, \quad \tilde{z} = \begin{bmatrix} \tilde{z}_1 \\ \vdots \\ \tilde{z}_m \end{bmatrix} \]

• \( \tilde{r}_i = Q^T r_i \) - Corresponds to a DCT-II

• \( z_i = Q \tilde{z}_i \) - Corresponds to an IDCT-II
Fast Transform Solvers
Complete Algorithm

1. Partition RHS vector $r$ into $m$ sub-vectors $r_i$
2. Apply $DCT-II$ on each $r_i$ sub-vector and obtain $\tilde{r}$
3. Permute $\tilde{r}$ with permutation matrix $P$ and obtain vector $\tilde{r}^P$
4. Solve the $n$ decoupled tridiagonal systems and obtain vector $\tilde{z}^P$
5. Apply inverse permutation $P^T$ on vector $\tilde{z}^P$ and obtain vector $\tilde{z}$
6. Apply $IDCT-II$ on each $\tilde{z}_i$ sub-vector and obtain vector $z$
Fast Transform Solvers

• Close-to-optimal complexity $O(mn\log(n))$ for the solution of the linear system
  – Require a total of $m$ DCT-II, $m$ IDCT-II and the solution of $n$ tridiagonal systems

• Large potential for parallelism
  – DCT-II can be implemented through FFT
  – Tridiagonal system solution algorithms entail large parallelism

• Mapping onto parallel architectures provide further acceleration
Outline

• Introduction
• Theoretical Background
• Proposed Methodology
• Experimental Evaluation
• Conclusions
Proposed Methodology

• **FTCG** combines the PCG method with a problem-specific and highly-tuned preconditioner

• **Key Observation:** 2D power grid matrices can be approximated with Fast Transform-based matrices
  – Accelerates convergence

• Fast Transform-based preconditioners
  – Efficient Construction
  – Highly-parallel solution algorithm
  – Limited memory requirements
  – Efficient for transient analysis with variable time-step
Preconditioner Construction

- Approximate the original 3D irregular power grid with a 2D regular grid

1. Collapse the 3D to a 2D grid by neglecting vias resistance

2. Create regular 2D grid by averaging resistance values among grid rails of different layers
Preconditioner Construction

• Regular 2D grid is used for preconditioning

\[
M = \begin{bmatrix}
T_1 & -g_1^v I \\
-g_1^v I & T_2 \\
-g_2^v I & T_3
\end{bmatrix}
\]

\[
T_1 = g_1^h \begin{bmatrix}
1 & -1 \\
-1 & 2 \\
-1 & 1
\end{bmatrix} + \left( g_1^v + \frac{c_1}{h} \right) I
\]

\[
T_2 = g_2^h \begin{bmatrix}
1 & -1 \\
-1 & 2 \\
-1 & 1
\end{bmatrix} + \left( g_1^v + g_2^v \right) I
\]

\[
T_3 = g_3^h \begin{bmatrix}
1 & -1 \\
-1 & 2 \\
-1 & 1
\end{bmatrix} + g_2^v I
\]

\(M\) and \(T_i\) matrices have a special form that allows utilization of a Fast Transform solver
Proposed Simulation Algorithm

- Use MNA to formulate the linear system $Ax = b$ for power grid simulation
- Use PCG with the proposed preconditioning mechanism for the solution of the linear system
- Offers ample task- and data-level parallelism
  - PCG comprises matrix-vector and vector-vector operations
  - The preconditioner solve step can be solved with the highly parallel Fast Transform solver
Outline

• Introduction
• Theoretical Background
• Proposed Methodology
• Experimental Evaluation
• Conclusions
Experimental Evaluation

• Intel Core 2 Quad @ 2.5 GHz, 8GB of main memory
• NVIDIA GeForce GTX 560 Ti GPU, 1GB of main memory
• Compared four simulation algorithms:
  – CHOLMOD, state-of-the-art direct solver
  – ICCG, multi-core implementation of PCG with Incomplete Cholesky Preconditioner
  – FTCG-CPU, multi-core implementation of FTCG
  – FTCG-GPU, GPU implementation of FTCG
• Industrial (IBM) and synthetic benchmarks with size ranging between 127K to 6.29M nodes
Convergence Rate
IBM Benchmarks

ICCG
FTCG

5.4X 4.7X 6.8X 6.9X 5.6X 4.4X 4.6X 5X

ICCG
FTCG

127K 851K 953K 10.7M 1.67M 1.46M 1.46M 2.62M

22/11/2012 ICCAD 2012
Convergence Rate
Synthetic Benchmarks

ICCG
FTCG

30X 31.5X 56.9X 73.4X 56.1X 72.6X
525K 1.04M 2.09M 3.14M 4.19M 6.29M
IBM Benchmarks

Execution Time

CHOLMOD
ICCG
FTCG-CPU
FTCG-GPU

113.2X 94.7X 137.6X 330.8X 164.9X 214.4X
22.1X 26.8X
19.7X 44.7X
12.8X 18.8X
851K 10.7M 1.67M 1.46M 1.46M 2.62M

22/11/2012
ICCAD 2012
Execution Time
IBM Benchmarks

- CHOLMOD
- ICCG
- FTCG-CPU
- FTCG-GPU

851K: 12.1X, 72.7X
10.7M: 9.5X, 70.4X
1.67M: 12X, 84.1X
1.46M: 10.5X, 77.4X
1.46M: 11.2X, 83.7X
2.62M: 17.3X, 138.8X

22/11/2012 ICCAD 2012 29
Execution Time
Synthetic Benchmarks

- CHOLMOD
- ICCG
- FTCG-CPU
- FTCG-GPU

N/A

22/11/2012
ICCAD 2012
Execution Time Synthetic Benchmarks

- CHOLMOD
- ICCG
- FTCG-CPU
- FTCG-GPU

N/A
Memory Requirements

- FTCG reduces memory requirements
  - Only the eigenvalues of $T_i$ matrices must be stored for the preconditioner solve step

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>$M_{CHOLMOD}$</th>
<th>$M_{ICCG}$</th>
<th>$M_{FTCG}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.62M</td>
<td>3350 MB</td>
<td>311 MB</td>
<td>239 MB</td>
</tr>
<tr>
<td>4.19M</td>
<td>7700 MB</td>
<td>615 MB</td>
<td>471 MB</td>
</tr>
<tr>
<td>6.29M</td>
<td>N/A</td>
<td>919 MB</td>
<td>685 MB</td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Theoretical Background
• Proposed Methodology
• Experimental Evaluation
• Conclusions
Conclusions

- **FTCG**: An efficient algorithm for analysis of large-scale power delivery networks found in nano-scale ICs
- Achieves significant acceleration in the simulation process and can harness the computational capabilities of parallel architectures
  - Highly-tuned preconditioner
  - Large degree of multi-grain parallelism
  - Computational and memory requirements that scale linearly with the power grid size
- **Future Work**:
  - Extend the preconditioner to take into account vias resistances
  - Power grid analysis on heterogeneous architectures
Acknowledgements

- Konstantis Daloukas is partially supported by Bodossaki Public Benefit Foundation
Questions?