B-028

ストリーミングSIMD拡張命令を用いた電子回路シミュレータSPICE3の高速化
Speed up of Circuit Simulator SPICE3 by Streaming SIMD Extentions

篠塚　研太 †　中村　あすか †　富永　浩文 †　前川　仁孝 †
Kenta Shinozuka　Asuka Nakamura　Hirobumi Tominaga　Yoshitaka Maekawa

1. はじめに

電子回路の特性解析を行うために SPICE3(Simulation Program with Integrated Circuit Emphasis 3) シミュレータが広く利用されている．回路特性は, 非線形連立微分方程式となる回路方程式を求解することで解析する. 連立方程式求解時に扱う係数行列は, ランダムスパース行列となる．SPICE3 は, ランダムスパース行列を用いた連立方程式を, 外積形式ガウス法を用いて求解する. 連立方程式求解時間は, シミュレーション時間の約 9 割を占める [1]. このため, シミュレーション時間の削減に, 連立方程式求解の高速化が重要となる.

行列演算を高速化する方法として，ストリーミング SIMD 拡張命令を用いた研究が行われている [2]．ストリーミング SIMD 拡張命令は, 近年の汎用プロセッサの多くに搭載されており，一命令で複数のデータを同時に処理する．SPICE3 は, 使用するメモリ量を削減するために, 零要素を除去した係数行列を作成する [3]. 係数行列内の要素数とストリーミング SIMD 拡張命令のベクトル長が一致しない場合, ベクトル化効率が低下し, 並列性が最大限に活用できない. ベクトル化効率は, 係数行列に演算とは無関係の要素を挿入し, 要素数とストリーミング SIMD 拡張命令のベクトル長が一致することで向上する．そこで本稿では, ストリーミング SIMD 拡張命令のベクトル化効率の向上により, SPICE3 シミュレーションを高速化する手法を提案する.

2. SPICE3 による連立方程式求

SPICE3 は, LU 分解法の手法の一つである外積形式ガウス法を用いて連立方程式を求解する. 係数行列 A の更新要素を a_{ij} とすると, 外積形式ガウス法で行う演算は式 (1), 式 (2) となる.

$$a_{ij} = (a_{ij} - \sum a_{ik} \times a_{kj})/a_{ii} \quad (i < j) \qquad (1)$$

$$a_{ij} = a_{ij} - \sum a_{ik} \times a_{kj} \quad (i \geq j) \qquad (2)$$

SPICE3 は, スパースな係数行列を扱うため, 係数行列をリスト構造として生成し, 行方向および列方向のリストにアクセスすることで非零要素を抽出する. SPICE3 で用いる外積形式ガウス法による LU 分解処理を図 1 に示す. 外積形式ガウス法による LU 分解を行う際, まず, 図中の対角要素 a_{ii} を基点とし, 行方向リストにアクセスすることで, 行方向リストから非零要素 a_{ik} を抽出し, $a_{ik} = a_{ik}/a_{ii}$ を計算する. この計算を, 行方向リスト内に存在する全ての非零要素に対して行う. 次に, a_{ii} を基点として, 列方向リストにアクセスすることで, 列方向リストから非零要素 a_{ik} を抽出し, 更新要素 a_{ij} に対して $a_{ij} = a_{ij} - a_{ik} \times a_{kj}$ を計算する. 更新終了後, a_{ik} から列方向リストにアクセスする. 同様の処理を列方向リストの末端まで実行する. その後, a_{kj} から行方向リストにアクセスし, 行方向リストの末端まで処理が終了後, 一つ右下の対角要素を基点とし, 同様の処理を行う.

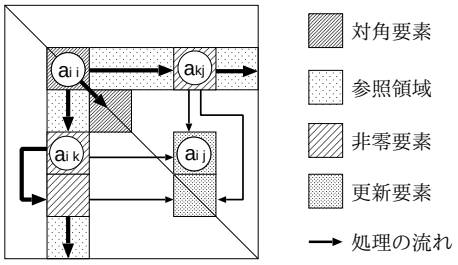

図 1: SPICE3 で行う LU 分解処理

3. SIMD 命令による並列化

ストリーミング SIMD 拡張命令は, 演算対象となるデータをビット長が 128 ビットの SIMD 専用レジスタに転送することで, 演算を同時に実行する．演算対象となるデータが単精度浮動小数点値の場合は演算を 4 並列化, 倍精度浮動小数点値の場合は演算を 2 並列化できる.

3.1 LU 分解の並列化

外積形式ガウス法による連立方程式求解は, 演算の並列性を抽出し, ストリーミング SIMD 拡張命令を用いることで並列に実行する．式 (1) および式 (2) における参照要素 a_{ik}, a_{kj}, a_{ii} は, 演算中に更新されることがない. そのため, 更新要素 a_{ij} に対し, a_{ii} を用いた除算, および a_{ik}, a_{kj} を用いた積差演算を並列に実行することができる．図 2 に外積形式ガウス法による LU 分解処理の並列化例を示す. 外積形式ガウス法よる LU 分解処理では, まず, 図中の対角要素 a_{ii} を基点とし, 行方向リストにアクセスする. このとき, a_{ii} をストリーミング SIMD 拡張命令のベクトル長分 SIMD レジスタに転送する. 行方向リストから要素を抽出する際, ループアンローリングを行い, 非零要素 a_{kj}, a_{kj+1} を SIMD レジスタへ転送する. SIMD レジスタに転送した要素に対して, $a_{kj}/a_{ii}, a_{kj+1}/a_{ii}$ をストリーミング SIMD 拡張命令を用いて同時に計算する. 除算終了後, a_{ii} を基点として列方向リストにアクセスして, 列方向リストから非零要素 a_{ik} を抽出し, a_{ik} をベクトル長分 SIMD レジスタに転送することで, $a_{ik} \times a_{kj}, a_{ik} \times a_{kj+1}$ を同時に計算する. 次に, 更新要素 a_{ij}, a_{ij+1} を SIMD レジスタに転送し, 乗算結果を用いて, a_{ij}, a_{ij+1} に対する減算を同時に実行する. 更新終了後, a_{ik} から列方向リストにアクセスし, 同様の処理を列方向リストの末端まで

†千葉工業大学情報工学科, Department of Computer Science, Chiba Institute of Technology

行う. その後, a_{kj+1} から行方向リストにアクセスし, ループアンローリングを行い, 除算処理を行方向リストの末端まで行う. この一連の処理を行列サイズまで繰り返すことで, 外積形式ガウス法による LU 分解を並列に実行する. 行方向リストに存在する非零要素 a_{kj}, a_{kj+1} は, 更新処理時に複数回参照する可能性があるため, SIMD レジスタ上にデータを残す. SIMD レジスタからデータを参照することで, メモリアクセス回数を削減し, 演算処理を高速化する.

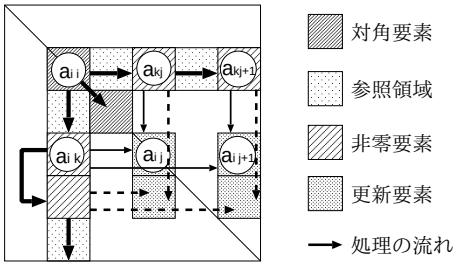

図 2: 外積形式ガウス法の並列化例

3.2 ベクトル化効率の向上方法

ループアンローリングを行う際, 行方向リスト内に並列に処理できる要素が存在しない可能性がある. そのため, データ転送時に条件分岐命令を用いて, 後続リスト内の非零要素の有無を判断する必要がある. 問題規模の増加に伴い，分岐回数が増加するため，演算時間が増加する. そこで, 条件分岐命令数を削減することで, 求解時間の高速化を行う. 条件分岐命令を削減するために, 係数行列内に零要素を挿入し, 演算可能な要素数とストリーミング SIMD 拡張命令のベクトル長が常に一致するように係数行列のデータ構造を変更する. 零要素を除算または積差演算に用いた場合, 乗算結果, 除算結果は零となるため, 本来の演算結果に影響を与えることがない. 係数行列への零要素の挿入方法として, まず対角要素を基点に, 行方向リストおよび列方向リストの非零要素数をカウントする. 次に, 非零要素数がベクトル長と一致しない場合, 零要素を末端リストとして追加する. これにより, 演算に用いる要素数とベクトル長が常に一致し, データ転送時に必要な条件分岐命令を除去する.

4. 評価

提案する手法の有効性を評価するため，従来の SPICE3 と SIMD 化を行った SPICE3 の過渡解析におけるシミュレーション時間を比較する．CPU は SIMD 演算が可能な Intel Core2 Duo E6600 を用いる．SPICE3 の連立方程式求解は, ストリーミング SIMD 拡張命令を用いて, 倍精度浮動小数点演算による 2 並列化で実行する．評価回路は, 行列サイズが 10×10 の 2 入力 NAND 回路と行列サイズが 179×179 の 4bit 加算器回路を用いる. 表 1 に, 各 SPICE3 のシミュレーション時間を示す. なお, 以下の評価では, SPICE3 は従来の SPICE3 の処理時間を表し, SIMD 化は SIMD 化した SPICE3 の処理時間を表す.

表 1 より, SIMD 化した SPICE3 のシミュレーション時間は従来の SPICE3 と比べ, 約 10% 高速化することが確認できた. 高速化の要因について詳しく評価するために, 処理時間が増加するモデル化計算時間と SIMD 化により処理時間が減少する行列演算時間を計測する．表 2 に, 従来の SPICE3 と SIMD 化を行った SPICE3 のモデル化計算時間を, 表 3 に, 従来の SPICE3 と SIMD 化を行った SPICE3 の行列演算時間を示す．表 2 より, SIMD した SPICE3 のモデル化計算時間が, 従来の SPICE3 より約 1.1% 増加することが確認できた. これは, SIMD 化を行った SPICE3 では, モデル化計算時に係数行列内に零要素を入力するため, 係数行列への零要素挿入時間がオーバヘッドとなり, シミュレーション時間に影響を与えたためであると考えられる. また, 表 3 より, SIMD 化した SPICE3 では, 従来の SPICE3 と比べ, 行列演算が約 11% 高速化することが確認できた. 零要素の挿入により増加するモデル化計算時間は SIMD 化による行列演算時間の削減に比べ, モデル化計算時に増加する時間は少なく, SPICE3 シミュレーションを高速化できることが確認できた.

表 1: シミュレーション時間の比較 [s]

回路名	SPICE3	SIMD 化
NAND 回路	8.31	7.82
4bit 加算器回路	261.66	249.32

表 2: モデル化計算時間の比較 [s]

回路名	SPICE3	SIMD 化
NAND 回路	0.96	0.98
4bit 加算器回路	34.73	37.65

表 3: 行列演算時間の比較 [s]

回路名	SPICE3	SIMD 化
NAND 回路	7.31	6.72
4bit 加算器回路	225.02	209.42

5. おわりに

本稿では, ストリーミング SIMD 拡張命令のベクトル化効率を向上させることにより, SPICE3 のシミュレーション時間を削減する手法を提案して, 評価を行った. 評価の結果, 従来の SPICE3 のシミュレーション時間に比べ, SIMD 化を行った SPICE3 のシミュレーション時間は約 10% 高速化することができた.

参考文献

[1] N.Kapre, A.Dehon:"Parallelizeng Sparse Matrix Solve for SPICE Circuit Simulation using FPGAs,"IEEE International Conference on Fiele-Programmable Technology, December 9-11, 2009.

[2] D.Aberdeen, J.Baxter:"Emmerald:A Fast Matrix-Matrix Multiply Using Intel's SSE Instructions,"Concurrency and Computation:Practice and Experience, August 26, 2000.

[3] 三浦 道子, 名野 隆夫, 盛 健次 :" 回路シミュレーション技術と MOSFET モデリング,"31 March 2003.