Vol.2015-SLDM-170 No.46
Vol.2015-EMB-36 No.46
2015/3/7

GPU を用いた LSI 基板の 3 次元発熱・伝熱シミュレータの構築と評価

渡邊聖剛[†1]　大村崇[†2]　北川友貴[†2]　林磊[†2]　孟林[†1]　福井正博[†1]

近年，回路の高集積化に伴う発熱の増大により，熱設計が重要となっている．熱設計を行う解決策に熱解析シミュレーションがあるが，計算量が膨大となる．我々はこれまで GPU を用いた 2 次元熱回路網モデルによる高速伝熱シミュレータを実現した．しかし，2 次元熱回路網モデルでは垂直方向の熱分布の解析精度が不十分である．
本研究では 3 次元での熱分布を考慮した 3 次元熱回路網モデルを提案する．既存の回路行列圧縮手法を 3 次元での解析に向けて拡張し，GPU を用いた 3 次元回路の解析計算の効率化を図った．その結果 3 次元の伝熱解析の高精度化を実現した．さらに，GPU の並列演算により，CPU のみと比べて最大約 14.6 倍の高速化を達成した．

A Three-dimensional Thermal Generation and Thermal Transfer Simulator for LSI Substrate by GPU

SHOUGO WATANABE[†1]　TAKASHI OMURA[†2]　YUUKI KITAGAWA[†2]
LAI LIN[†2]　LIN MENG[†1]　MASAHIRO FUKUI[†1]

Recently, thermal design is important by the increased heat generated with the high integration of circuits. There is a thermal analysis simulation for the solutions, but the amount of calculation is huge and time consuming. We have realized a high-speed thermal transfer simulator by two-dimensional thermal network model which uses a GPU. But analysis accuracy of thermal distribution in the vertical direction is insufficient.
In this study, we propose a three-dimensional thermal network model considering the thermal distribution in three-dimensional. And we tried the efficiency of analytical calculation of the three-dimensional circuit using GPU by extension of existing circuit matrix compression approach for the analysis of three-dimensional. As a result, we have achieved improving the accuracy of analyzing the three-dimensional thermal transfer. Furthermore, we have achieved about 14.6 times speed-up than a CPU program while accelerating the speed of calculation by using GPU parallel computing.

1. はじめに

近年，電子機器の高性能化に伴う回路の高集積化や，電池のエネルギー高密度化によって発熱の問題が顕著になってきている．そのため，製品設計の段階から熱の問題を考慮する熱設計が重要となっている．熱設計を行う手段の一つに熱解析シミュレーションがあるが，計算量が膨大となってしまう．そこで GPU の並列演算を汎用計算に用いて計算の高速化を図る．

GPU による熱解析シミュレーションについて，大村らの熱回路網法を用いた GPU によるシミュレーションの高速化の研究[1]がある．これは林らの電源配線シミュレーションの高速化手法[2]を，熱解析シミュレーションに応用したものである．解析対象を規則的な熱回路を用いて表現し，電源配線回路と同様の規則性に基づくものであることを利用して，高速な熱解析シミュレーションを実現した．しかし，3 次元の伝熱を回路の縮約によって，2 次元の回路モデルで表現していたため，3 次元の熱分布を解析するには不十分であった．本研究では，より高精度な解析を行うため，3 次元の熱回路網モデルを提案する．また提案モデルに適合するように従来の手法を拡張し，GPU を用いて処理の高速化を図る．

2. 熱解析シミュレーション

2.1 熱回路網モデル

本研究の解析対象は熱回路網法[3]で表現する．熱回路網法は，熱抵抗を用いて熱回路網を作り，回路解析手法を使って温度を求める方法である．ここで用いる熱抵抗とは，ノード間の熱の伝わりにくさであり，R_T [K/W]で表わす．また物体の伝熱をノードの温度 T [K]，ノード間の熱の流れを熱流量 Q_T [J/S]と表わし，これらを用いて伝熱経路を表現する．さらに，ノードに温度あたり蓄熱できる量は熱容量 C_T [J/K]，熱量は H [J]で表される．熱回路におけるパラメータは，電気回路におけるパラメータの関係と相似である．よって電気回路と同様に熱回路においても同様にオームの法則や過渡応答等が成立する．表 1 は熱と電気のパラメータの対応を表わす．

†1 立命館大学　理工学部
College of Science and Engineering, Ritsumeikan University.
†2 立命館大学　理工学研究科
Graduate School of Science and Engineering, Ritsumeikan University

ⓒ2015 Information Processing Society of Japan

Vol.2015-SLDM-170 No.46
Vol.2015-EMB-36 No.46
2015/3/7

表 1　熱と電気の各パラメータ対応

Table 1　Correspondence between electronic and of parameters

熱回路		電気回路	
熱抵抗	$\boldsymbol{R_T}$[K/W]	電気抵抗	$\boldsymbol{R_E}$[Ω]
温度	$\boldsymbol{T}$[K]	電圧	$\boldsymbol{E}$[V]
熱流量	$\boldsymbol{Q_T}$[J/S]	電流	$\boldsymbol{I}$[A]
熱容量	$\boldsymbol{C_T}$[J/K]	静電容量	$\boldsymbol{C_E}$[F]
熱量	$\boldsymbol{H}$[J]	電荷量	$\boldsymbol{Q_E}$[C]

本研究における熱回路網モデルを図 1 に示す．3 次元領域をメッシュ分割し，分割された 1 個の領域ごとに回路網の 1 個のノードを対応させる．解析対象の分割数が熱回路網のノード数に対応する．各ノードには各分割領域の発熱を表わす熱流源 Q_0 と，熱容量 C が接続される．またノード間には各領域に対応した熱抵抗が接続される．この熱回路網は大村らの 2 次元熱回路と同様にノードが格子状に並ぶ規則性を持つ．

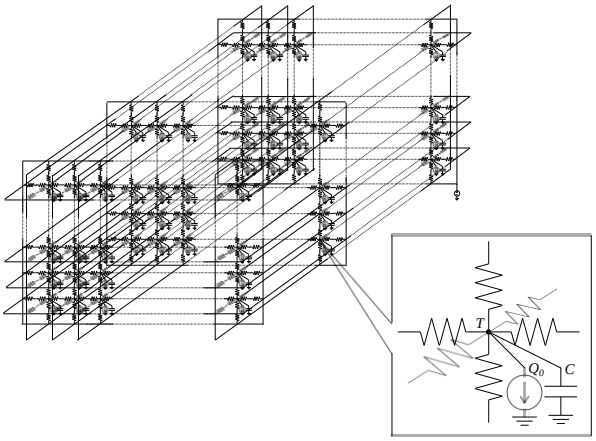


図 1　熱回路網モデル

Figure 1　Thermal network model

3 次元の熱回路網モデルにおける端は空気と接するため，端のノードには空気の熱抵抗が接続される．熱抵抗を求めるにあたり熱伝導率が必要となるが，空気は温度によって熱伝導率が変わる.そのため用いる熱伝導率を随時更新し，熱抵抗を書き換える必要がある．本研究において，空気の熱伝導率λaは山本らの式(1)を用いて更新する[4].

$$\lambda a = -3.9 \times 10^{-8} T_{amb}{}^2 + 1.01 \times 10^{-4} T_{amb} - 5.55 \times 10^{-4} \quad (1)$$

ここでの温度T_{amb}には，壁面温度と無限遠の流体温度の平均である膜温度を使用する．この熱伝導率λaを用いて空気の熱抵抗θ_aを求める．これは式(2)で表わされる．

$$\theta_a = \frac{1}{\lambda \mathrm{a}} \times \frac{L}{\mathrm{S}} \quad (2)$$

ここでの L は空気のブロックの長さ，S は空気のブロックの断面積を示す．このようにして熱抵抗を書き換える．

2.2 熱回路解析手法

熱回路網の各ノードにおいて，温度に対する微分方程式を解くことで解析を行う．解法として後退オイラー法を用いる．後退オイラー法は時間に関する微分項について後退差分を用いて近似する．収束性に優れ，一般的な回路シミュレータにも用いられる．Δt が 0 より大きい，微小な時間刻み幅であるとき, 時刻 t+Δt における微分項について考えると，式(3)のようになる．

$$\frac{dh(t+\Delta t)}{dt} = \lim_{\Delta t \to 0} \frac{h(t+\Delta t) - h(\Delta t)}{\Delta t} \approx \frac{h(t+\Delta t) - h(t)}{\Delta t}, \quad \Delta t > 0 \quad (3)$$

これを整理すると式(4)のように表わせる．

$$h(t+\Delta t) = h(t) + \Delta t \dot{h}(t+\Delta t), \quad \Delta t > 0 \quad (4)$$

この斬化式より解を求める．

式(4)を用いてあるノード(i,j,k)の温度$T^{(\mathrm{i,j,k})}$を導く. 温度$T^{(\mathrm{i,j,k})}$は熱量$H^{(i,j,k)}(t)$とノードの熱容量$C^{(i,j,k)}$を用いて，式(5)で表すことができる．

$$T^{(\mathrm{i,j,k})} = \frac{H^{(i,j,k)}}{C^{(i,j,k)}} \quad (5)$$

熱量$H^{(i,j,k)}(t)$ はノードの熱容量$C^{(i,j,k)}$に流れる熱流量$Q_all^{(i,j,k)}(t+\Delta t)$ によって決まる．これを式(6)に示す．

$$H^{(i,j,k)}(t+\Delta t) = H^{(i,j,k)}(t) + Q_al^{(i,j,k)}(t+\Delta t) \times \Delta t \quad (6)$$

$Q_all^{(i,j,k)}(t+\Delta t)$ は, あるノード(i,j,k)の 6 方向から流出入する熱流量と定熱流源$Q_0^{(i,j,k)}$ によって決まる．ここで$Q_all^{(i,j,k)}(t+\Delta t)$はノードにおける熱量の単位時間当たりの変化に等しい．すなわち $\dot{H}^{(i,j,k)}(t+\Delta t)$である．よって式(6)は式(4)に示される後退オイラー法による近似計算にほかならない．さらに，ノード間の熱抵抗をRとすると$Q_all^{(i,j,k)}(t+\Delta t)$は式(7)で表わせる．

$$Q_{all}^{(i,j,k)}(t+\Delta t) = \sum_{\substack{(x,y,z)=(i-1,j,k),(i+1,j,k),(i,j-1,k),\\(i,j+1,k),(i,j,k+1),(i,j,k-1)}} \frac{T^{(\mathrm{i,j,k})}(t+\Delta t) - T^{(\mathrm{x,y,z})}(t+\Delta t)}{R_{(i,j,k)-(x,y,z)}} \quad (7)$$

式(6)に式(5)と式(7)を代入し整理すると，式(8)となる．

$$\frac{\Delta \mathrm{t}}{\mathrm{C}}\left[-\frac{1}{\mathrm{R}}-\frac{1}{\mathrm{R}}-\frac{1}{\mathrm{R}}-\frac{1}{\mathrm{R}}-\frac{1}{\mathrm{R}}-\frac{1}{\mathrm{R}}\frac{6\Delta \mathrm{t}+\mathrm{CR}}{\Delta \mathrm{tR}}\right]\begin{bmatrix} T^{(i-1,j,k)}(t+\Delta t) \\ T^{(i+1,j,k)}(t+\Delta t) \\ T^{(i,j-1,k)}(t+\Delta t) \\ T^{(i,j+1,k)}(t+\Delta t) \\ T^{(i,j,k+1)}(t+\Delta t) \\ T^{(i,j,k-1)}(t+\Delta t) \\ T^{(i,j,k)}(t+\Delta t) \end{bmatrix} = T^{(\mathrm{i,j,k})}(t) - \frac{\Delta \mathrm{t}}{\mathrm{C}} Q_0 \quad (8)$$

全てのノードに対して式(8)によって温度を求めるため，全体の計算は線形連立一次方程式に帰着する．また今回の熱回路網は格子状であるため，ノード間の接続は規則的である．その規則性から係数行列は対角優位となる．

ⓒ2015 Information Processing Society of Japan

A)　ヤコビ法

後退オイラー法を提案する熱回路網モデルに適用すると，線形連立一次方程式の計算に帰着する．この線形連立一次方程式を解くためにヤコビ法を用いる．ヤコビ法は行列の反復計算を行うことで線形連立一次方程式の解を求める緩和法の一つである．今回求まる線形連立 1 次方程式の係数行列は対角優位であるため，ヤコビ法の計算は収束することが保証される[5]．またヤコビ法は，各計算にデータの依存性がなく，並列化に適している．一般的に数値解析に用いられる式は式(9)である．

$$x_p^{(m)} = \frac{1}{a_{pp}}\left(b_p - \sum_{q \neq p}^{N} a_{pq} x_q^{(m-1)} \right) \qquad (9)$$

ここでa_{pq}，b_pは p，q を行列の番号として，それぞれ左辺の行列成分と右辺のベクトル成分に対応する．またx_pは求める解のベクトルの成分である．$x^{(m)}$は初期解 x^0 に対し m 回目の反復で得られる値とする．ε を最大許容誤差とするとき，ヤコビ法のアルゴリズムは式(10)のように表わせる．

$$\sum_{m}^{N} \left| x_p^{(m)} - x_p^{(m-1)} \right| \leq \varepsilon \qquad (10)$$

式(10)を満たさない限り，式(9)の反復を行う．よって方程式の解は，$x_p^{(m)} - x_p^{(m-1)}$の差の絶対値の合計が最大許容誤差より小さくなったときの$x_p^{(m)}$である．

B)　CSCG (Compressed Sparse for Cubic Grid structure)

後退オイラー法による熱解析手法ではメモリの使用量が膨大となるため，GPU にアルゴリズムを実装する際，使用データ量の削減が必要となる．そこで CSCG (Compressed Sparse for Cubic Grid structure)を用いて方程式の回路行列データの圧縮を行う．これは大村らが 2 次元熱回路網モデルに用いていた疎行列圧縮手法 CSG(Compressed Sparse for Grid structure)を，3 次元熱回路網モデルで使用できるよう拡張したものである．CSG は 2 次元熱回路網において，元の行列における各ノードの成分を表す対角成分の値，上下左右の接続を表す非対角成分の値，非対角成分の列位置をそれぞれ行列に分け，元の行列データを表現する．これは 2 次元熱回路網のノードが上下左右 4 ノードとの接続に限られる規則性のためである．しかし，3 次元熱回路網では上下左右に加え上層下層の計 6 つの隣ノードとの接続を考慮する必要がある．そのため CSCG では CSG の保持データに加え，上層下層を含めた計 6 つの隣ノードのデータとインデックスを保持する．これにより，2 次元のモデルと同様に回路行列の圧縮が可能である．回路上での行番号と列番号，層番号をそれぞれ i，j，k とし，水平方向の領域分割数を N とする．CSCG による表現では各ノードについて，図 2 のように情報が保存される．

各ノードデータ

$$\left[a_{iN+j+k(N\times N),\ iN+j+k(N\times N)} \right]$$

隣ノード接続データ

上	$\left[a_{iN+j+k(N\times N),\ (i-1)N+j+k(N\times N)} \right]$
下	$\left[a_{iN+j+k(N\times N),\ (i+1)N+j+k(N\times N)} \right]$
左	$\left[a_{iN+j+k(N\times N),\ iN+j-1+k(N\times N)} \right]$
右	$\left[a_{iN+j+k(N\times N),\ iN+j+1+k(N\times N)} \right]$
上層	$\left[a_{iN+j+k(N\times N),\ iN+j+(k+1)(N\times N)} \right]$
下層	$\left[a_{iN+j+k(N\times N),\ iN+j+(k-1)(N\times N)} \right]$

隣ノード接続データインデックス

上	$[(i-1)N+j+k(N\times N)]$
下	$[(i+1)N+j+k(N\times N)]$
左	$[iN+j-1+k(N\times N)]$
右	$[iN+j+1+k(N\times N)]$
上層	$[iN+j+(k+1)(N\times N)]$
下層	$[iN+j+(k-1)(N\times N)]$

図 2　CSCG 保持データ

Figure 2　Stored data on CSCG

CSCG では回路行列データの圧縮に加え，各成分のデータのインデックスと値を直接利用できるので，効率良く処理を行える．CSCG での各データの検索手順は図 3 のようになる．

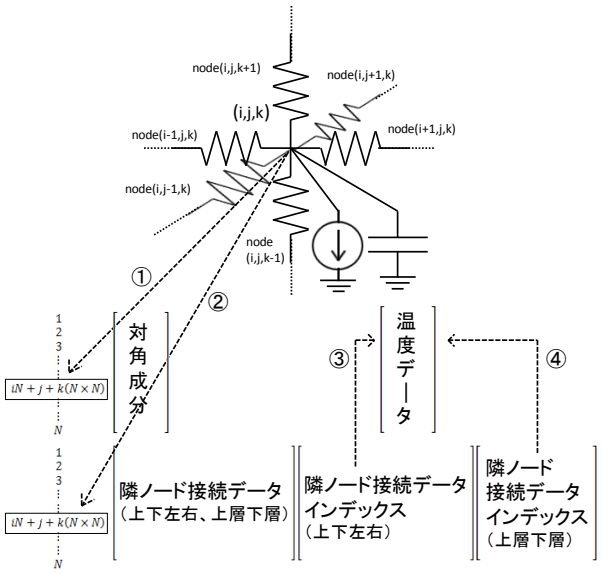


図 3　CSCG 法のメモリ対応関係

Figure 3　Correspondence between the memory on CSCG

回路の各頂点(i, j, k)に対応した行列のノード番号は n = iN + j+ kN^2 である．①まず各ノードの成分を表わす対角成分 d_nとb_nへは直接参照できる．②またノード n に隣接する非対角成分の接続データ，およびそのデータのインデックス

も同様に直接参照できる．③，④さらに，接続するノードの温度$x_*^{(m-1)}$は，隣ノード接続データインデックスから参照できる．よってメモリアクセスの高速化を図れる．ここで対角優位疎行列の高速圧縮手法の先例である DIA 形式[6]と比較を行う． DIA 形式を 3 次元熱回路網モデルにおけるヤコビ法の計算に用いた場合，頂点(i, j, k)から$x_q^{(m-1)}$とb_pの参照を行うために，乗算 18 回，加算 15 回，減算 3 回，共有メモリへの参照 14 回と，計 50 回の手数が必要である．しかし CSCG では共有メモリへの参照 20 回だけと，DIA 形式に比べて約 40%の手数で済む．ヤコビ法が式(9)の反復計算を必要とすること，隣ノードのより少ない 2 次元熱回路網モデルに対し，参照が必要なデータが増えていることから，本手法によるデータアクセスの効率化は有用である．

3. 実装アルゴリズム

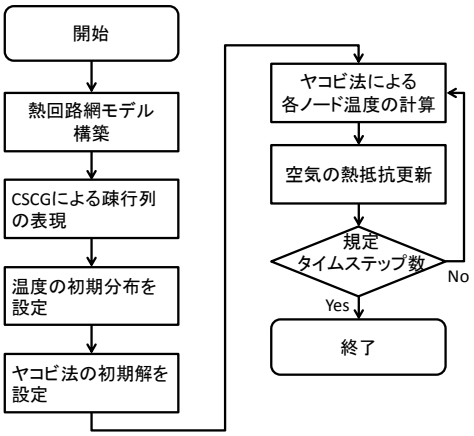


図 4　処理フロー図
Figure 4　Process flow

図 4 に回路シミュレーションにおける全体の処理の流れ図を示す．処理の流れとしては，まず解析する熱回路網モデルの構築，すなわち解析する熱回路網モデルの回路行列データを生成する．次に CSCG を用いて回路行列データの圧縮を行う．その後温度の初期分布，ヤコビ法計算の初期解を設定する．これらのデータを元に，ヤコビ法を用いた演算と熱抵抗の更新を規定タイムステップ数行い，Δt 後の頂点ノードの温度 $x(t+\Delta t)$を導く．

GPU への実装部分は図 5 のように表わされる．各ノードでのヤコビ法による計算などの並列処理可能な部分を GPU 側で実行し，各種の判定といった処理は CPU 側で行う．まずヤコビ法の式(9)を用いて$x^{(m-1)}$から$x^{(m)}$を導出する．次に式(10)を用いて算出誤差の総和を求める．この二つの手続きを，式(10)の最大許容誤差 ε 以下に収まるまで順次繰り返す．その後空気の熱抵抗を更新し，次の演算に移る．このとき GPU で必要となるデータは CPU のホストメモリからグローバルメモリに書き込み，用いる．GPU での各処理でグローバルメモリから CSCG のデータを取りこみ，共有メモリを用いて内部で計算を行う．共有メモリへのアクセスはヤコビ反復計算 1 回につき 20 回である．したがってアクセス回数は，20×総反復回数となる．

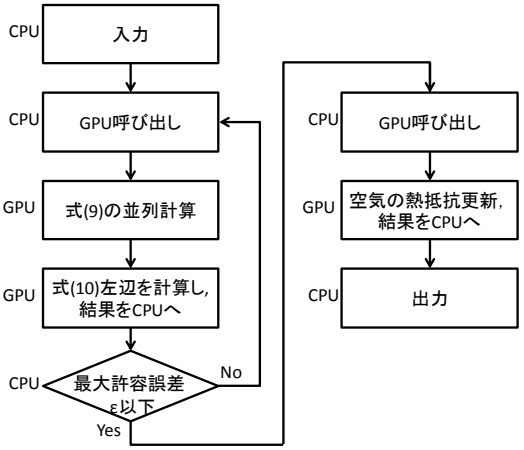


図 5　ヤコビ法実装アルゴリズム
Figure 5　An algorithm for implementing Jacobi method

4. 実験結果

4.1 実験条件

解析対象は一般的な LSI パッケージを想定し，国峰らの文献[7]に示された多層モデルを採用する．層を 10 層とし，1～3 層を基板部，4～5 層をはんだボール部，6～10 層を部品部として，図 6 の物理モデルに対してシミュレーションを行う．発熱は第 8 層で起きている．同物理モデルのパラメータは表 2 に示す．空気の熱伝導率については，(1)式を用いる．シミュレーションを行うに当たって，過渡応答における時定数から十分小さい値を後退オイラー法のタイムステップΔt [s]として設定する必要がある．今回の回路モデルは RC 回路である．RC 回路の時定数は R×C であり，今回はこれを 1/10 してタイムステップとする．これを式で表わすと，式(11)になる．

$$\Delta t = \frac{R \times C}{10} \tag{11}$$

ここで用いる熱抵抗 R[K/W]はシミュレーションに用いる熱抵抗の中で一番小さいもの，熱容量 C [J/K]は全体の熱容量を熱回路サイズで割った，ノード毎の熱容量である．

熱回路サイズをそれぞれ 16×16×10，32×32×10，64×64×10 とした場合に対して，CPU のみによる実行と，GPU による並列処理を用いた場合との速度比較を行う．ハードウェア環境は，CPU は Intel Core i7 960 (3.20 GHz)を，GPU は NVIDIA GeForce GTX 580 (512 Core, 1.54 GHz)を使用する．コーディング環境は CPU のプログラムは共に Visual Studio 2008，GPU の並列処理は NVIDIA 社の CUDA(Compute Unified Device Architecture) [8]を用いる．

ⓒ2015 Information Processing Society of Japan

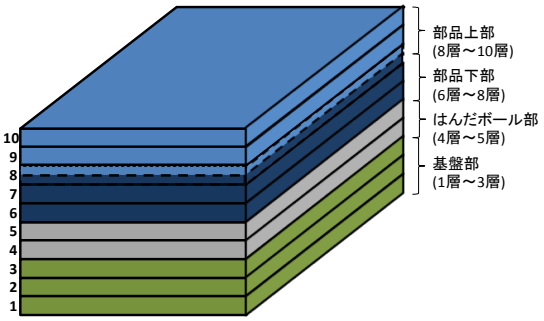


図 6　物理モデル

Figure 6　Physical model

表 2　解析条件

Table 2　analysis condition

物理サイズ (L×W×H) [cm]	部品上部	3.0×3.0×0.075
	部品下部	3.0×3.0×0.075
	はんだボール部	3.0×3.0×0.06
	基板部	3.0×3.0×0.09
	空気(垂直方向)	3.0×3.0×1.0
	空気(水平方向)	1.0×3.0×0.3
熱伝導率 [W/(m・K)]	部品上部	1.0
	部品下部(垂直/水平)	0.25 / 30.0
	はんだボール部(垂直/水平)	15.5 / 0.027
	基板部	0.16
全体の熱容量[K/W]		3.0
周囲気温[K]		300.0

4.2　実験結果

図 7 は熱回路サイズを 64×64×10 としたときの，10 秒後のシミュレーション結果である．左から順に 1～5 層，5～10 層である．左上が最下層，右下が最上層となる．

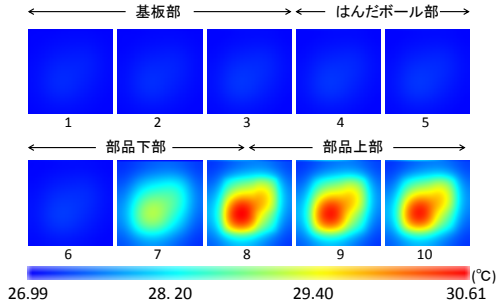


図 7　シミュレーション結果（64×64×10）

Figure 7　Simulation results (64×64×10)

表 3 は熱回路サイズを 16×16×10，32×32×10，64×64×10 とした場合の，10 秒後の結果をシミュレートするのにかかった時間を示す．表 3 のデータを基に計算した，GPU を用いた場合の速度向上比率のグラフを図 8 に示す．横軸は回路サイズ，縦軸は GPU 実行時間と CPU 実行時間を比べた場合の速度向上比率である．回路サイズが 64×64×10 では約 14.5 倍の高速化の効果が得られた．より大きい回路サイズに対しては，並列化の恩恵が大きくなるため，さらに大きな高速化率が見込まれる．

表 3　シミュレーション実行時間（3 次元）

Table 3　Simulation runtime (three-dimensional)

回路サイズ	CPU	GPU
16×16×10	22.636[s]	5.975[s]
32×32×10	89.434[s]	7.738[s]
64×64×10	386.598[s]	26.566[s]

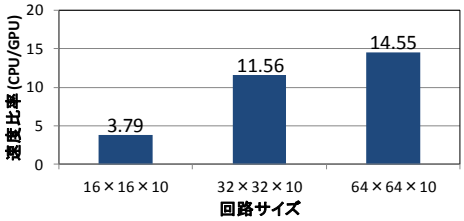


図 8　速度向上比率

Figure 8　Speed-up ratio

表 4 は GPU を用いた場合の総反復回数と，総反復回数から求めた共有メモリへのアクセス回数を示す．共有メモリを用いる場合，それぞれ独立したアドレス空間から同一のメモリ空間にアクセスできる．アクセスにあたってカーネルを経由する必要がなく，速度上の利益が見込める．共有メモリを用いることができなかった場合，この大量なアクセス分をグローバルメモリで行う必要がある．これは大きなオーバーヘッドを生むため，反復計算における共有メモリの利用は実行速度に大きく寄与していると考えられる．

表 4　総反復回数と共有メモリへのアクセス回数

Table 4　The total number of iterations and the number of accesses to the shared memory

回路サイズ	総反復回数	共有メモリへのアクセス回数
16×16×10	46500	930000
32×32×10	51488	1029760
64×64×10	144153	2883060

4.3　2 次元熱回路網モデルとの比較

大村らの 2 次元熱回路網モデル[1]との比較を行う．まずシミュレーション実行時間の比較を行う．同解析条件で熱回路サイズをそれぞれ 16×16，32×32，64×64 とした場合の実行時間を表 5 に示す. 今回表 5 の結果では GPU の処理時間の方が遅くなってしまった．回路サイズが小さい時には，全体の処理に対して CPU と GPU の通信時間の割合が大きいため高速化があまり見込めない．今回の比較にあた

ⓒ2015 Information Processing Society of Japan

って与えた実験条件下ではそれが顕著であった．表 3 と表 5 を比べると，全ての場合において 3 次元熱回路モデルの方の時間が増えている．これは 3 次元に変更したことにより，計算量が増えたためであると考えられる．まず層が 10 層に増え，ノード数が 10 倍になっているため，全体の計算が増えている．またノード毎に接続される熱抵抗の数が 4 本から 6 本に増え，ノード毎の計算の重みも増えている．さらにタイムステップも，計算時間が増えた要因の一つである．タイムステップは回路の時定数にしたがって決める必要がある．回路サイズが大きくなる，すなわち，分割数が増えると，分割領域における熱抵抗と熱容量が小さくなる．そのためタイムステップがより小さく設定される．タイムステップが小さくなると計算の数が増えるため，時間も増えたと考えられる．空気の熱抵抗を計算毎に更新することも要因の一つである．

表 5　シミュレーション実行時間（2 次元）
Table 5　Simulation runtime (two-dimensional)

回路サイズ	CPU	GPU
16×16	0.010[s]	0.359[s]
32×32	0.110[s]	0.484[s]
64×64	1.251[s]	1.560[s]

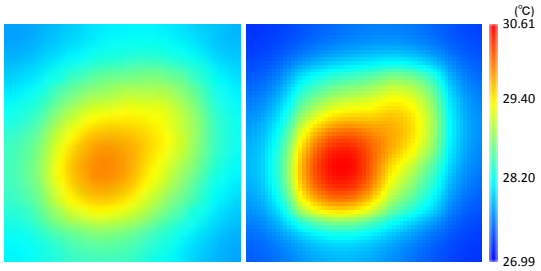


図 9　シミュレーション結果（64×64，図 7 の第 8 層）
Figure 9　Simulation results(64×64，Eighth layer in Figure 7)

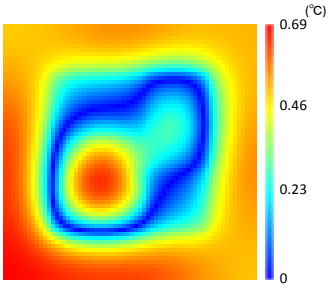


図 10　温度差（絶対値）
Figure 10　Temperature difference (Absolute value)

次にシミュレーション結果の出力画像についての比較に移る．図 9 の左側は同解析条件で熱回路サイズを 64×64 としたときの，10 秒後のシミュレーション結果，右側は比較に用いる図 7 の第 8 層の拡大図である．また図 10 は，図 9 の左右を比較したときの，温度差の絶対値を視覚化したものである．図 10 の温度差は，最大約 0.694 度であった．図 9 を見てみると，右図の方が熱源中心は温度が高く，外側は温度が低くなっている．さらに温度の分布を見てみると，左図では全体の温度差が小さいため，熱の勾配が緩やかであるとわかる．2 次元熱回路網モデルでは今回提案する 3 次元熱回路網モデルに比べ，少ないノード数で垂直方向の熱の伝搬が簡略化されていた．垂直方向の熱分布を考慮すると，水平方向に熱の伝わりやすい部品下部へ熱が伝わるまで時間がかかる．2 次元回路モデルではこれが考慮されておらず，発熱ノードからの水平方向の伝熱経路が部品上部と下部全体として直接設定されていた．そのため水平方向の伝熱が本来より大きく計算され，熱の拡散が早く進む解析結果として表われたと考えられる．このことから，特に熱伝導率に異方性のあるモデルでは 3 次元での熱分布を考慮することが重要であることが確認できる．

5. まとめ

LSI の 2 次元熱解析モデルを 3 次元に拡張し，GPU を用いた 3 次元熱解析シミュレーションを実現した．実際にシミュレーションを行い，熱分布のシミュレーション結果画像の生成と，シミュレーション実行時間の速度検証を行った．さらにシミュレーション結果について，2 次元熱解析モデルにおける結果と比較し，考察した．今回の実験では熱回路サイズについては最大で 64×64×10 まで検証を行った．そして CPU のみによる実行と比べ，GPU を用いた場合に約 14.5 倍の高速化を実現した．今後，より大きな回路サイズについても検証を行う．また，シミュレーション結果の精度の確認のため，同一モデルの別のシミュレーション結果との比較が必要であると思われる．

参考文献

[1]　大村崇，林磊，孟林，福井正博，“GPGPU を用いた VLSI チップ熱解析の一手法”，電子情報通信学会技術研究報告.VLD, VLSI 設計技術，vol.113，pp.209-214，Nov 2013
[2]　林磊，塩野隼人，福井正博，“GPGPU による電源回路シミュレーションに対する後退オイラー法の実装と評価”，電子情報通信学会論文誌 A，Vol.J96-A No.5 pp.215-224，May 2013.
[3]　国峰尚樹，エレクトロニクスのための熱設計完全入門，溝口 勲夫，日刊工業新聞社，東京，1997.
[4]　Y. Yamamoto, K. Kato, L. Lin, and M. Fukui, "A cool charger for lithium-ion battery," in Proc. 18th Workshop on Synthesis and System Integration of Mixed Technologies, Sapporo, Japan, Oct. 2013.
[5]　浅井秀樹，渡邉貴之，電子回路シミュレーション技法，村越勝弘，科学技術出版，東京，2003.
[6]　N. Bell and M. Garland, “Efficient sparse matrix-vector multiplication on CUDA,” NVIDIA Technical Report, NVR-2008-004, 2008.
[7]　国峰尚樹，電子機器の熱流体解析入門，日刊工業新聞社，東京，2009.
[8]　J. Sanders and E. Kandrot,　株式会社クイープ（翻訳），「CUDA by Example 汎用 GPU プログラミング入門」，インプレスジャパン，東京，　2011.

ⓒ2015 Information Processing Society of Japan