FDTD法による電磁界シミュレータの開発とGPGPUによる高速演算

生産技術室　城之内 一茂，名和 礼成

A design of FDTD EM simulator,
and high-speed computing by GPGPU technology

Kazushige JOUNOUCHI and Yukinari NAWA

電磁波の振る舞いを可視化し，その理解を直感的に行えることを目的に，電磁界解析の一手法であるFDTD法を用いたプログラムを作成した。これらを手軽に・自由に使えるようフリーなライセンスのもと，ソースコードと実行バイナリ，及び幾つかの条件設定ファイルのサンプルを一般に公開した。

また，CUDAを用いたGPGPUによる高速演算の可能性を示し，そのアーキテクチャと効率的な使用方法について検討を行った。

１．はじめに

電磁波は目に見えないため，その振る舞いを直感的に理解することは難しい。これは，配線のアンテナとしての振る舞いや意図しないループの形成，反射による重ね合わせなどによる電磁波の放射やノイズ耐性の低下等，回路の高周波的な振る舞いへの理解が難しいということでもある。

本研究では，直感的に電磁波の振る舞いに対する理解を助けるツールを手軽に・自由に使用できることを目的に，電磁界解析の一手法であるFDTD法[1)]を用いて電磁波の振る舞いを計算し，実行環境に依存しないOpenGL[2),3)]を用いて可視化を行い，これを広く一般に公開した。

また，CUDAを用いたGPGPUによる高速演算の可能性を示し，そのアーキテクチャと効率的な使用方法について検討を行った。

２．FDTD法による電磁界の可視化

２．１ FDTD法

FDTD法は，解析空間を細かな立方体に区切り，その辺上に電界，面上に磁界を配置したYeeセルと呼ばれる微小領域(図１)を用いて解析を行う。ここで，空間的に交互に配置された電解・磁界を時間的に交互に演算することにより，マクスウェルの方程式を逐次計算するものである。

解析空間のモデル化は，Yeeセルの電界・磁界が配置されている辺・面上に，その空間及び隣接空間にある物質の物性（誘電率・透磁率・導電率等）により計算される係数を適用することにより表現される。このため，様々な物質が配置された解析空間も，セルの細かさによる制限はあるものの，複数の物性をもった物体が存在するような解析も容易に行うことが可能である。

FDTD法は逐次計算法であるため，ノイズの放射や伝搬といったノイズの発生メカニズムのような過渡現象を観察することが可能である。また，解析領域に適切な境界条件を設定することにより，自由空間での解析も可能である。

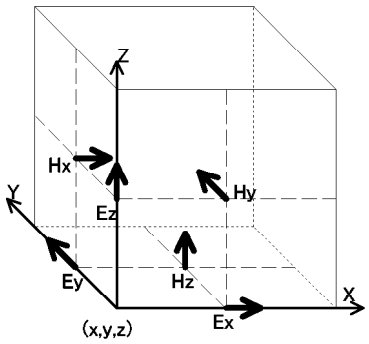

図１ Yeeセル

２．２ 電磁界シミュレータの設計

今回の開発においては，電磁波の振る舞いを手軽に・直感的に理解するのに最低限必要な要素として，以下のとおりの仕様とした。

・解析空間の条件設定，媒質の物性値，オブジェクトのモデリング等，解析要件はテキストファイルにより記述できる
・オブジェクトの形状は基本形状である「線」「面」「立方体」とする

・解析空間の境界条件は完全導体による閉空間「PEC」若しくは自由空間を摸す吸収境界条件「Liao(２次)」とする
・画面出力はXY・YZ・ZXの各平面に電界または磁界の強さを色で表現し，演算結果を逐次出力することによりアニメーション表現を行う
・マウス操作により，各出力平面の切り出し位置を移動可能とし，出力の拡大縮小・回転・移動等が行えることとする

また，必要に応じて仕様を追加できるよう，内部的に以下の仕様を盛り込んでいる。

・周波数分散性媒質として，RC法による一次のデバイ分散性媒質の取扱い
・MPIによる並列演算や，解析空間の分割による効率的な解析空間の配置が出来るよう，複数の解析領域を持てる

ここで，出力の表現(可視化)手法として，計算プラットフォームに依存せず，移植性の高いOpenGLを用いた。OpenGLでは，様々なプラットフォームで同様に使用可能なフリーの補助ライブラリ群が多数提供されており，これを用いることでウインドウの生成等，環境に深く依存する処理をもプラットフォームに依存せずに開発することが可能となる。図２に解析例を示す。

今回開発したプログラムは自由に使え，必要な改変が自由に出来るよう，フリーなライセンスのもと，ソースコードの公開を行った。また，手軽に実行できるよう，実効バイナリ(Windows)及び幾つかの条件設定ファイルのサンプルを共に公開した。

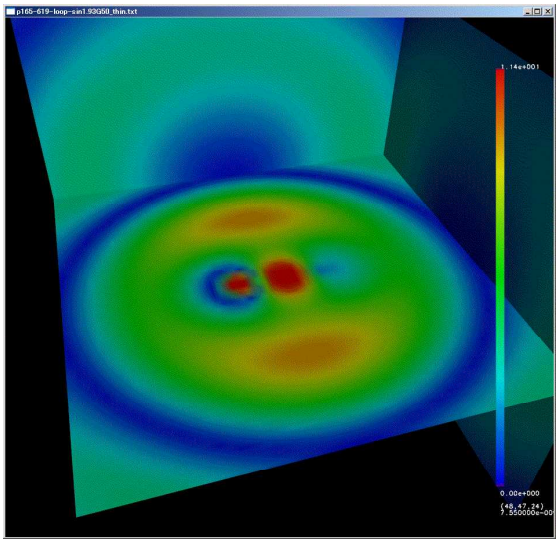

図２ 解析例

３．GPGPUによる高速演算

近年，三次元CGによるアプリケーションの要求が高まるなか，画像演算装置(GPU)の性能は飛躍的に向上した。この進化の過程で，それまで特定の機能毎にハードウェアで演算回路を構成されていたものが，プログラム可能な並列演算器によるソフトウェア演算へと技術がシフトしてきた。そのような中，GPUを並列積和演算装置として汎用演算に使用する技術(GPGPU)が注目されている。

今回は，C言語を拡張した言語体系を持つ開発環境であるNVIDIA社のCUDA[4]を用いてその特性と性能を検証した。

３．１ CUDAデバイス

CUDAデバイスをハードウェアから見ると，最低実行単位であるSP(Streaming Processor)と呼ばれる実効ユニットを複数個まとめたSM(Streaming Multiprocessor)を１つの実行単位とし，並列演算を行うものである。ソフトウェアから見ると，１つの演算処理を複数のBlockに分け，その中で複数のThreadが展開される。

両者の関係は，BlockとSMの単位で関係づけられており，各SM内のSPは，同じ命令を並列に実行することとなる。このため，SMは一つのSIMD演算器(Single Instruction Multiple Data)と見ることができる。

メモリアーキテクチャとしては，低速だが大容量であり，また値の保持が可能なGlobal Memory，高速だが値の変更が不可能なConstant Memory，高速だが小容量であり，またブロック内でしか共有できないShared Memory等がある。

このような構成と高速なタスクスイッチとにより遅いメモリアクセスを隠蔽し，高度な並列演算を行っている。このため，メモリへのアクセス順序・単位・位置・密度が演算速度に大きく影響し，いかに(プロセッサにとって)良いメモリアクセスを行うかが高いパフォーマンスを得るための重要な要素となっている。

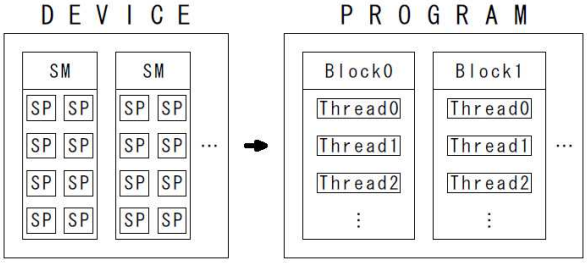

図３ CUDAアーキテクチャ

３．２ CUDAによる並列演算

今回行った演算速度の比較に使用した環境を以下に，各計算デバイスとその使用技術，演算結果を表１に示す。

- ・GPU　NVIDIA　QuadroFX380(16SP)
- ・CPU　Intel　XeonX3350(QUADcore)
- ・メモリ　Trancend　DDR2-800(4GBytes)
- ・OS　Microsoft　WindowsXP(32bit)
- ・開発環境 Microsoft VisualC++2008
- ・CUDA環境 NVIDIA　CUDAToolkit(3.0)
- ・ドライバ Developer Drivers 197.13
- ・演算対象 2次元FDTD (8層PML吸収境界)

CUDA-malloc(cudaMalloc関数による一次元の連続メモリ領域確保)の項(250×250)と(256×256)を比較すると，単なる連続領域の確保では解析領域の大きさと演算時間とで逆転が起きている。しかし，CUDA-mallocPitch(cudaMallocPitch関数による二次元領域確保)の項では逆転は起きない。両者の違いはGlobal Memoryへのメモリ空間の取得関数の違いだけである。このことからプロセッサが如何に効率的にメモリへアクセスできることが演算速度に重要か，即ちメモリアクセスが如何にコストが高く，また効率的なアクセスの条件が限定的であるかがわかる。

CUDA-SharedMemoryの項は，cudaMallocPitch関数によるメモリ領域の確保を行ったうえで，高速なShared Memoryをキャッシュメモリとして用いている。これにより，遅いGlobal Memoryへのアクセスの低減と，メモリアクセスを直線的で効率的な順序で行うことを可能としている。

GOLDの項目はCPUによる演算を示している。ここで，OpenMPの項では，QuadCoreCPUを用いているが，Thread数が２で最速であったため2Threadの値を用いている。

両デバイスの最速値(CUDA-SharedMemory速度，GOLD-OpenMP+SSE)を比較すると，解析領域2048×2048においてほぼ同じ性能となった。

４．おわりに

FDTD法による電磁界シミュレータを作成し，条件ファイルのサンプル，バイナリ及びソースコードを公開した。これにより，回路の高周波的な振る舞いを手軽に・自由に可視化することが出来た。

CUDA環境におけるGPGPUによる高速演算の可能性については，今回使用した最もローエンドなデバイス(16SP)においてもCPUでの演算とほぼ同等の性能となった。GPUはCPUと並列演算が可能であるため，演算とその他の周辺処理を分けて行うことにより，このようなローエンドの製品でもその利用価値は十分高い。GPUの得意とする積和演算は信号処理等との親和性が高い技術であり，著者たちは，ロボット制御への応用を検討している。

参考文献

1)宇野亨,FDTD法による電磁界およびアンテナ解析,コロナ社,1998

2)Khronos Group,OpenGL-The Industry Standard for High Performance Graphics, http://www.opengl.org/

3)床井浩平,GLUTによる「手抜き」OpenGL入門, http://www.wakayama-u.ac.jp/~tokoi/opengl/libglut.html

4)NVIDIA,CUDA Zone,http://www.nvidia.co.jp/object/cuda_home_jp.html

表１　演算結果[sec]

計算デバイス及び利用技術		解析領域サイズ				
		250×250	256×256	512×512	1024×1024	2048×2048
CUDA	Malloc	3.4	1.4	5.4	21.5	85.5
	MallocPitch	0.9	1.0	3.8	14.0	55.0
	SharedMemory	-	0.9	3.3	12.4	48.5
GOLD	none	-	2.7	8.9	20.5	62.3
	SSE	-	1.6	5.4	17.0	59.3
	OpenMP	-	1.9	6.8	15.2	51.9
	OpenMP+SSE	-	1.1	4.5	13.2	49.8