IMPLEMENTATION OF THE FINITE-DIFFERENCE TIME-DOMAIN METHOD USING GRAPHICS PROCESSING UNITS

Approved by:

---

Dr. Marc Christensen

---

Dr. Nathan Huntoon

---

Professor Ira Greenberg

IMPLEMENTATION OF THE FINITE-DIFFERENCE TIME-DOMAIN

METHOD USING GRAPHICS PROCESSING UNITS

A Thesis Presented to the Graduate Faculty of the

Lyle School of Engineering

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Master of Electrical Engineering

with a

Major in Electrical Engineering

by

S. David Lively

(B.S.E.E, Southern Methodist University, 2008)

August 1, 2016

## ACKNOWLEDGMENTS

I thank my committee for their patience, insight and unfailing encouragement. Without them, this thesis would remain vaporware. Never give up, never surrender!

Lively , S. David B.S.E.E, Southern Methodist University, 2008

Implementation of the Finite-Difference Time-Domain
Method Using Graphics Processing Units

Advisor: Professor Marc Christensen
Master of Electrical Engineering degree conferred August 1, 2016
Thesis completed August 1, 2016

Traditionally, optical circuit design is tested and validated using software which implement numerical modeling techniques such as Beam Propagation, Finite Element Analysis and FDTD.

While effective and accurate, FDTD simulations require significant computational power. Existing installations may distribute the computational requirements across large clusters of high-powered servers. This approach entails significant expense in terms of hardware, staffing and software support which may be prohibitive for some research facilities and private-sector engineering firms.

Application of modern programmable GPGPUs to problems in scientific visualization and computation has facilitated dramatically accelerated development cycles for a variety of industry segments including large dataset visualization, microprocessor design, aerospace and electromagnetic wave propagation in the context of optical circuit design.

The FDTD algorithm as envisioned by its creators maps well to the massively-multithreaded data-parallel nature of GPUs. This thesis explores a GPU FDTD implementation and details performance gains, limitations of the GPU approach, optimization techniques and potential future enhancements that may provide even greater benefits from this underutilized and often-overlooked tool.

## TABLE OF CONTENTS

LIST OF FIGURES ........ vii

CHAPTER

1. INTRODUCTION ........ 1
   - 1.1. FDTD Overview ........ 2
     - 1.1.1. Wave equation ........ 2
     - 1.1.2. Yee Cell ........ 2
2. DEVICE ARCHITECTURE ........ 5
   - 2.1. CPU ........ 5
   - 2.2. GPU ........ 5
     - 2.2.1. SIMD ........ 5
     - 2.2.2. FDTD in SIMD ........ 5
3. MEEP ........ 6
   - 3.1. Background ........ 6
   - 3.2. Modeling approach ........ 6
   - 3.3. Performance ........ 6
   - 3.4. Usability ........ 6
4. GOLIGHTLY ........ 7
   - 4.1. goals ........ 7
   - 4.2. system architecture ........ 7
     - 4.2.1. Host ........ 7
     - 4.2.2. GPU ........ 7
   - 4.3. Modeling approach ........ 7

4.4. Implementation . . . . . 7
4.5. Testing methodology . . . . . 7
4.5.1. Test Model . . . . . 7
4.5.2. Analytical Result . . . . . 7
4.5.3. Numerical Result . . . . . 7
4.5.4. Comparison . . . . . 7
4.6. Additional Examples . . . . . 8
4.6.1. Coupler . . . . . 8
4.6.2. Splitter . . . . . 8
5. Conclusions . . . . . 9
5.1. Meep performance . . . . . 9
5.2. GoLightly performancec . . . . . 9
5.3. Meep vs GoLightly . . . . . 9
5.4. Results . . . . . 9
5.5. Limitations . . . . . 9
6. FUTURE WORK . . . . . 10
APPENDIX
REFERENCES . . . . . 11

## LIST OF FIGURES


Figure | Page

1.1. 2D $TM_Z$ Yee Cell ........ 3

*To Audrey, Wyatt, Walter and Gwendolyn*

# Chapter 1

# INTRODUCTION

FDTD [1] is a proven algorithm, first published in (...) by (yee, et al). It is the underlying mechanism used by many commercial optics simulation packages, as well as open source software such as MIT's Meep.

Given the computationally-intensive nature of FDTD, organizations requiring simulation of large domains or complex circuits must provide significant resources. These may take the form of leased server time or utilization of an on-site high-performance cluster, amongst other options.

In this thesis, we explore an implementation of the Finite-Difference, Time-Domain (FDTD) method of electromagnetic waves simulation as implemented on graphics processing units (GPUs). Initially designed to perform image generation tasks such as those required by games, cinema and related fields, modern versions are well-suited for general computation work. GPUs are now enjoying wide adoption in fields such as machine learning and artificial intelligence, medical research, signals analysis and other areas which require rapid analysis of large datasets.

Even modern consumer-grade GPUs offer thousands or tens of thousands of processing units, while high-end CPUs offer 4-8 cores. While the two are not interchangeable (see: chapter on Device Architecture), some algorithms, such as FDTD, require little or data interdependence, no branching logic (a severe performance impediment on GPUs) and consist of short cycles of simple operations. The power of the GPU lies in performing these simple operations at large scale, with thousands of threads running in parallel.

The following sections detail FDTD. Later sections describe a CPU-based implementation (MIT's Meep simulator), and our GPU-based GoLightly simulator. We verify the GPU solution numerically, and compare performance between CPU- and GPU-based implementations. Finally, we consider future applications and enhancements.

## 1.1. FDTD Overview

At it's heart, FDTD expresses Maxwell's equations as a discretized set of time-domain equations. These equations describe each electric field component in terms if its orthogonal, coupled magnetic fields, and each magnetic field component as a function of its coupled, orthogonal electric fields.

### 1.1.1. Wave equation

In a $TM_z$ time domain simulation, wave equation for $E_z$ is of the form:

$$\frac{\partial E_z}{\partial t} = K * (\frac{\partial H_x}{\partial y} + \frac{\partial H_y}{\partial x}) \tag{1.1}$$

Equation 1.1 states that the temporal derivative (change in amplitude) of $E_z$ is a function of the $Y$-axis spatial derivative of the $H_x$ field and the $X$-axis spatial derivative of the $H_y$ field.

In order to apply this equation to a computational domain, FDTD defines a cell-based discretization strategy.

### 1.1.2. Yee Cell

Yee [1] defines a computational unit known as a "cell." The cell describes how each field component within a domain is related to it's coupled fields. For instance, in a 2D $TM_z$ simulation, $E_Z$ depends on adjacent $H_y$ and $H_x$ components. The cell format used in such a simulation is of the form shown in 1.1.

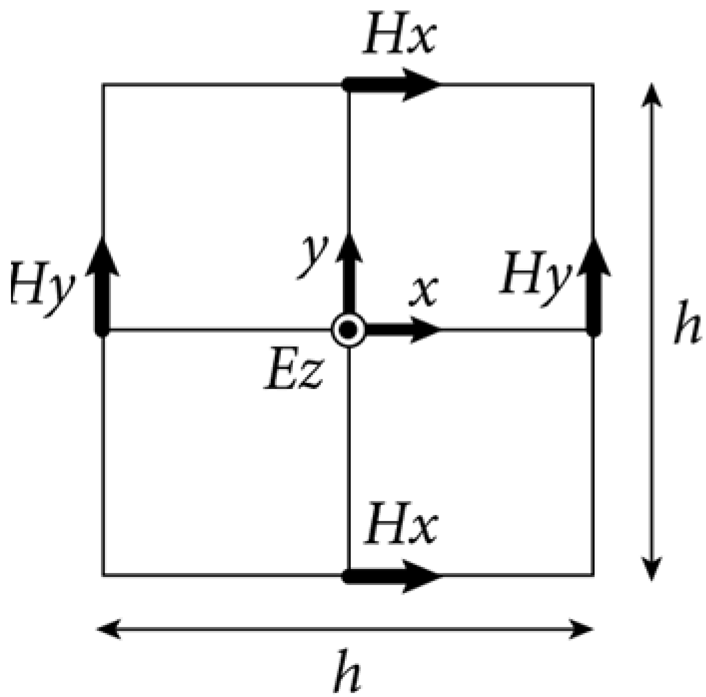


Figure 1.1. 2D $TM_Z$ Yee Cell

More formally, we may expand the $E_Z$ wave equation, arriving at:

$$E_{z\,i,j}^{\,t} = C_a * E_{z\,i,j}^{\,t-1} + C_b * (H_{x_{i,j+\frac{1}{2}}}^{\,t-\frac{1}{2}} - H_{x_{i,j-\frac{1}{2}}}^{\,t-\frac{1}{2}}) + C_b * (H_{y_{i+\frac{1}{2},j}}^{\,t-\frac{1}{2}} - H_x y_{i-\frac{1}{2},j}^{\,t-\frac{1}{2}}) \tag{1.2}$$

Similarly, the equations for the coupled fields $H_x$ and $H_y$ may be expressed as:

$$H_{x\,i,j}^{\,t} = D_a * H_{x\,i,j}^{\,t-1} + D_b * (E_{z_{i,j+\frac{1}{2}}}^{\,t-\frac{1}{2}} - E_{z_{i,j-\frac{1}{2}}}^{\,t-\frac{1}{2}}) \tag{1.3}$$

$$H_{y_{i,j}}^{\,t} = D_a * H_{y_{i,j}}^{\,t-1} + D_b * (E_{z_{i+\frac{1}{2},j}}^{\,t-\frac{1}{2}} - E_{z_{i-\frac{1}{2},j}}^{\,t-\frac{1}{2}}) \tag{1.4}$$

where

content...

Chapter 2

# DEVICE ARCHITECTURE

## 2.1. CPU

independent cores, separate cache, dedicated ALU and registers

## 2.2. GPU

### 2.2.1. SIMD

SIMD - single ALU for multiple register sets

### 2.2.2. FDTD in SIMD

why FDTD maps well to GPUs

# Chapter 3

# MEEP

## 3.1. Background

## 3.2. Modeling approach

## 3.3. Performance

## 3.4. Usability

Chapter 4

# GOLIGHTLY

## 4.1. goals

## 4.2. system architecture

### 4.2.1. Host

### 4.2.2. GPU

## 4.3. Modeling approach

## 4.4. Implementation

## 4.5. Testing methodology

### 4.5.1. Test Model

### 4.5.2. Analytical Result

### 4.5.3. Numerical Result

### 4.5.4. Comparison

## 4.6. Additional Examples

### 4.6.1. Coupler

### 4.6.2. Splitter

Chapter 5

# Conclusions

## 5.1. Meep performance

## 5.2. GoLightly performancec

## 5.3. Meep vs GoLightly

## 5.4. Results

## 5.5. Limitations

# Chapter 6

# FUTURE WORK

future work...

## REFERENCES


[1] Yee, K. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation 14*, 3 (1966), 302–307.