Contemporary Engineering Sciences, Vol. 3, 2010, no. 2, 87 - 96

Formalization of the Yee Algorithm Using HOL Theorem Prover

Walid J. Ghandour

American University of Beirut
Department of Electrical and Computer Engineering
Beirut, Lebanon
wjg00@aub.edu.lb

Abstract

Finite-difference time-domain (FDTD) is a very popular numerical method for the solution of problems in electromagnetic. It is a simple and elegant way to discretize the differential form of Maxwell's equations. It is based on Taylor's series expansion. In this paper, we present the formalization of the Yee algorithm using HOL theorem prover. This represents the first step in our formalization of FDTD. HOL theorem prover is an interactive theorem prover that has been used in the formal verification of both hardware and software and in the formalization of pure mathematics.

Keywords: FDTD, formal modeling, theorem proving, Yee algorithm

1 Introduction

Since the early days of the development of electrical devices, engineers and scientists have tried to improve their designs for economy and efficiency. New techniques were developed to model devices and predict their performance at the design stage. Hardware prototypes or scaled models were used at the early stages. This approach proved to be expensive, cumbersome, and time consuming. It can only measure terminal quantities due to nonavailability of sophisticated sensors and instrumentation. Field plotting techniques were then developed, such as conformal mapping and curvilinear squares. This was followed by experimental methods carried out with electrolytic tanks and conducting paper analogs. This approach also proved to be non reliable due to the effect of temperature and humidity variations on measurements. Analytical methods were then pursued with limitation to techniques applicable to

linear isotropic media, the entire half-space, or semi-infinite regions. It was only possible to use constant values for permeability, conductivity, and constant dielectric permittivity. The applicability of the techniques depends on the system of coordinate used. Modeling of non linear problem was only possible for simple problems where transformation methods could be used. New techniques were developed with the introduction of calculating machines (analog and digital computers). These methods are considered numerical methods.

FDTD is arguably the most commonly used numerical method. Several softwares were developed that implements FDTD. To the best of our knowledge, no work was previously done to formalize FDTD.

2 Electric and Magnetic Fields

2.1 Electric Fields

The electric charge is the source of all electromagnetic phenomena. Electrostatics is the study of electric fields due to charges at rest. In SI units, electric charge has the units of coulombs. The charge of one electron is $1.602 * 10^{-19}$ coulombs. An electric charge of Q coulombs produces an electric flux of Q coulombs. The total flux passing through any closed surface is equal to the total charge enclosed by that surface. The electric flux density $\vec{D}$ is defined as the local value of the flux per unit area.

$$\oint_S \vec{D} \, .d \, \vec{S} = \int_v \rho dv.$$

For a point charge

$$\vec{D} = Q/4\pi r^2$$

where r is the distance from the source point to the field point.

There is a force between electric charges that varies as $1/r^2$. The force is proportional to the magnitude of the charges and depends on the medium. This force is expressed using Coulomb's law as follow:

$$\vec{F} = Q_1 Q_2 / 4\pi\epsilon_0 r^2$$

where ϵ_0 is the permittivity of free space. The force on a unit charge is called the electric field and is denoted as $\vec{E}$. The units of electric field are volts/meter.

$$\vec{D} = \epsilon_0 \, \vec{E} \, .$$

It is often difficult to compute the field quantities $\vec{D}$ and $\vec{E}$ directly. All the algebra is vectorial and we need to find three components at each point. It is simpler to compute scalar variables and then use the obtained values to

compute the field. Electric potential V is used to compute the electric field. The electric potential at a point in space is defined as the work expended in moving a unit charge from infinity to that point.

$$V = -\int_{\infty}^{P} \vec{E} \,.d\, \vec{l} \ .$$

$$\vec{E} = -\nabla V.$$

2.2 Magnetic Fields

Single magnetic charges, monopoles, equivalent to electric charges have not been observed in nature. However, it is often convenient to use it from a computational point of view. A magnetic point charge of magnitude m, will produce m units of magnetic flux in the same way that an electric charge produces electric flux. In SI units, magnetic charge and magnetic flux have the units of webers. Magnetic flux density $\vec{B}$ from a point charge is defined as:

$$\vec{B} = (m/4\pi\epsilon_0 r^2)\hat{a}_r.$$

There is a force on the magnetic charge that depends on the medium and varies inversely as r^2. The magnetic field of a point charge in free space is defined as:

$$\vec{H} = (Q_m/4\pi\mu_0 r^2)\hat{a}_r$$

$$\vec{B} = \mu_0 \vec{H}$$

where μ_0 (henries/meter in SI units) is the permeability of free space.

3 Finite-Difference Time-Domain Method

The finite-difference time-domain (FDTD) method is a very popular numerical method for the solution of problems in electromagnetic. in 1966, Yee proposed the FDTD method as an elegant and simple approach to discretize differential Maxwell's equations [10]. FDTD Yee algorithm is second order accurate in both space and time. Its correct stability criteria was presented by Taflove [8]. The original Yee algorithm uses cartesian grids that approximate a surface with one that is staircased. Lot of research was done to analyse errors introduced by different gridding for different structures.

FDTD method can be used to obtain either scattered or total fields. Scattered fields are calculated in the exterior portion of the computational domain, while, total fields are calculated over an interior subsection.

An absorbing boundary condition (ABC) is used to truncate the computational domain when modeling open-region problems. This is due to the fact that tangential components of the fields along the outer boundary cannot be obtained using the FDTD method. ABCs are grouped under two types: differential equations ABCs and material absorber ABCs. Perfectly matched layer (PML) technique, material absorber ABC, provides better accuracy than most other techniques. It becomes a gold standard to which other ABC techniques are compared.

In material-based ABCs, the computational domain is surrounded with a lossy material that dampens the outgoing fields. PML ABC introduced by Berenger [1] in 1994 significantly advanced the usage of material ABCs. Berenger ensures that the electromagnetic-wave phase velocity is the same in both the PML region and free space for all frequencies and angles of incidence by introducing a split-field extension of Maxwell's equations. PML region can be terminated with a perfect electric conductor (PEC). The reflection from the PEC to the interior of the computational space are often insignificant. A perfectly matched layer is ideal in the continuous domain only. Discrete form implementation introduces numerical errors.

One of the major advantages of FDTD with respect to other numerical methods is its ability to obtain wideband results using transient excitation. This can be achieved by including frequency-dependent properties of the material such as permittivity, conductivity and permeability.

3.1 The Yee Algorithm

In 1966, Yee introduced a set of finite-difference equations for the time-dependent Maxwell's curl equations [10]. The algorithm continues to have great usefulness. Following are some of its characteristics:

1. The Yee algorithm solves for both electric (E) and magnetic (H) fields in time and space. Using both E and H information results in a more robust solution (an accurate solution for a wide class of structures). This also helps in modeling features unique to each field, such as tangential and looping H singularities and radial E singularities.

2. The Yee algorithm centers its E and H components in three-dimensional space. Every E (H) component is surrounded by four circulating H (E)components. Some of the attributes of the Yee space lattice are:

    - The finite-difference expressions are central-difference and second order accurate.
    - Gauss's law relations are implicitly enforced. The mesh is divergence free in the absence of free electric and magnetic charge.

- Tangentional E and H continuity is maintained across an interface of different materials that is parallel to one of the lattice coordinate axes.

3. The Yee algorithm centers its E and H components in time in a leapfrog arrangement. All of the E computations in the modeled space for a particular time point are obtained and stored using previously stored H data. H computations are then obtained and stored using the newly obtained E data. The cycle repeats until time-stepping is concluded. Leapfrog time-stepping is fully explicit and non-dissipative (numerical waves do not decay due to a non-physical artifact). New value of the magnetic or electric field vector depends on its previous value, previous values of the components of the other field vector at adjacent points, and electric and magnetic current sources.

The Cartesian Yee space lattice and its staggered leapfrog time-stepping algorithm have proven to be very flexible, accurate and robust for a variety of problems.

3.1.1 Presentation of Maxwell's Equations Using FDTD

The following system of scalar equations is equivalent to Maxwell's equations in the rectangular coordinate system (x,y,z), using the MKS system of units and assuming that the dielectric parameters are time independent:

$$\frac{\delta H_x}{\delta t} = \frac{1}{\mu}(\frac{\delta E_y}{\delta z} - \frac{\delta E_z}{\delta y}) \tag{1}$$

$$\frac{\delta H_y}{\delta t} = \frac{1}{\mu}(\frac{\delta E_z}{\delta x} - \frac{\delta E_x}{\delta z}) \tag{2}$$

$$\frac{\delta H_z}{\delta t} = \frac{1}{\mu}(\frac{\delta E_x}{\delta y} - \frac{\delta E_y}{\delta x}) \tag{3}$$

$$\frac{\delta E_x}{\delta t} = \frac{1}{\epsilon}(\frac{\delta H_z}{\delta y} - \frac{\delta H_y}{\delta z} - \sigma E_x) \tag{4}$$

$$\frac{\delta E_y}{\delta t} = \frac{1}{\epsilon}(\frac{\delta H_x}{\delta z} - \frac{\delta H_z}{\delta x} - \sigma E_y) \tag{5}$$

$$\frac{\delta E_z}{\delta t} = \frac{1}{\epsilon}(\frac{\delta H_y}{\delta x} - \frac{\delta H_x}{\delta y} - \sigma E_z) \tag{6}$$

Yee notates any space lattice point as

$$(i, j, k) = (i\delta, j\delta, k\delta)$$

and any function of space and time as

$$F^n(i,j,k) = F(i\delta, j\delta, k\delta, n\delta t)$$

where $\delta = \delta x = \delta y = \delta z$ is the space increment and δt is the time increment.

Yee uses finite difference expressions for the space and time derivatives that are second-order accurate in δ and δt.

$$\frac{\delta F^n(i,j,k)}{\delta x} = \frac{F^n(i+1/2,j,k) - F^n(i-1/2,j,k)}{\delta} + O(\delta^2)$$

$$\frac{\delta F^n(i,j,k)}{\delta t} = \frac{F^{n+1/2}(i,j,k) - F^{n-1/2}(i,j,k)}{\delta t} + O(\delta t^2)$$

Yee positions the components of $\vec{E}$ and $\vec{H}$ about a unit cell of the lattice and evaluates them at alternate half-time steps.

The result of these assumptions is the following system for the system of (1) - (6):

$$\begin{aligned}
&H_x^{n+1/2}(i,j+1/2,k+1/2) = \\
&H_x^{n-1/2}(i,j+1/2,k+1/2) + \tfrac{\delta t}{\mu(i,j+1/2,k+1/2)\delta} * \\
&\quad (E_y^n(i,j+1/2,k+1) - E_y^n(i,j+1/2,k) \\
&\qquad E_z^n(i,j,k+1/2) - E_z^n(i,j+1,k+1/2))
\end{aligned}$$

$$\begin{aligned}
&H_y^{n+1/2}(i+1/2,j,k+1/2) = \\
&H_y^{n-1/2}(i+1/2,j,k+1/2) + \tfrac{\delta t}{\mu(i+1/2,j,k+1/2)\delta} * \\
&\quad (E_z^n(i+1,j,k+1/2) - E_z^n(i,j,k+1/2) \\
&\qquad + E_x^n(i+1/2,j,k) - E_x^n(i+1/2,j,k+1))
\end{aligned}$$

$$\begin{aligned}
&H_z^{n+1/2}(i+1/2,j+1/2,k) = \\
&H_z^{n-1/2}(i+1/2,j+1/2,k) + \tfrac{\delta t}{\mu(i+1/2,j+1/2,k)\delta} * \\
&\quad (E_x^n(i+1/2,j+1,k) - E_x^n(i+1/2,j,k) \\
&\qquad + E_y^n(i,j+1/2,k) - E_y^n(i+1,j+1/2,k))
\end{aligned}$$

$$\begin{aligned}
&E_x^{n+1}(i+1/2,j,k) = \\
&(1 - \tfrac{\sigma(i+1/2,j,k)\delta t}{\epsilon(i+1/2,j,k)})E_x^n(i+1/2,j,k) + \tfrac{\delta t}{\epsilon(i+1/2,j,k)\delta} * \\
&(H_z^{n+1/2}(i+1/2,j+1/2,k) - H_z^{n+1/2}(i+1/2,j-1/2,k) \\
&\quad + H_y^{n+1/2}(i+1/2,j,k-1/2) - H_y^{n+1/2}(i+1/2,j,k+1/2))
\end{aligned}$$

$$\begin{aligned}
&E_y^{n+1}(i,j+1/2,k) = \\
&(1 - \tfrac{\sigma(i,j+1/2,k)\delta t}{\epsilon(i,j+1/2,k)})E_y^n(i,j+1/2,k) + \tfrac{\delta t}{\epsilon(i,j+1/2,k)\delta} * \\
&(H_x^{n+1/2}(i,j+1/2,k+1/2) - H_x^{n+1/2}(i,j+1/2,k-1/2) \\
&\quad + H_z^{n+1/2}(i-1/2,j+1/2,k) - H_z^{n+1/2}(i+1/2,j+1/2,k))
\end{aligned}$$

$$\begin{aligned}
&E_z^{n+1}(i,j,k+1/2) = \\
&(1 - \tfrac{\sigma(i,j,k+1/2)\delta t}{\epsilon(i,j,k+1/2)})E_z^n(i,j,k+1/2) + \tfrac{\delta t}{\epsilon(i,j,k+1/2)\delta} * \\
&(H_y^{n+1/2}(i+1/2,j,k+1/2) - H_y^{n+1/2}(i-1/2,j,k+1/2) \\
&\quad + H_x^{n+1/2}(i,j-1/2,k+1/2) - H_x^{n+1/2}(i,j+1/2,k+1/2))
\end{aligned}$$

4 HOL Theorem Prover

HOL is one of the descendants of LCF, a proof-checking program developed at Stanford University by Robin Milner in 1972. Robin Milner invented the LCF-approach to theorem proving, designed the ML (*metalanguage*) programing language underlying it, and the polymorphic type system used both by ML and the LCF and HOL logics. LCF abbreviates Logic for Computable Functions. It is a proof checker for Scott's logic and is described by Milner as follows [5]: "The proof-checking program is designed to allow the user interactively to generate formal proofs about computable functions and functionals over a variety of domains, including those of interest to the computer scientist - for example, integers, lists and computer programs and their semantics. The user's task is alleviated by two features: a subgoaling facility and a powerful simplification mechanism".

HOL abbreviates Higher Order Logic. It is a proof assistant for higher order logic originally created for hardware verification at the register transfer level. It utilizes the simple type theory of Church [3] along with Hindley-Milner polymorphism [5] to implement higher-order logic. The first version of HOL was created by modifying the parser, pretty-printer, primitive axioms and inference rules of the Cambridge LCF. The HOL system emphasizes definition rather than axiom postulation as the primary method of developing theories. HOL has a derived type-definition principle that converts description of recursive datatypes into primitive definition and then automatically derives the natural induction and primitive recursion principles for the datatype. This tool has helped in changing the perception of HOL from a purely hardware verification system to a general purpose proof assistant. The new HOL simplifier integrates rewriting, conversions and decision procedures.

HOL has a number of key features: a simple core logic, LCF-style full expansiveness, support for various proof styles and a large library of user-supplied theories and proof tools.

HOL has four separate kinds of primitive terms: variables, constants, function applications and λ-abstractions. Other notations such as infixes, conditional terms, set abstractions, restricted quantifications, tuple notation, and tupled variable binding are considered as derived forms.

All proofs are expanded into sequences of primitive inferences. HOL has eight primitive inference rules which are implemented using ML functions. Application of these rules is the only way to obtain theorems in the system. Proved theorems can be saved and used in future proofs.

Set of types, type operators, constants and axioms available in HOL are organized in the form of theories. There are two main primitive theories, *bool* and *ind*, for booleans and individuals (a primitive type to denote distinct elements), respectively. Theorems can be derived based on these two main

theories and added to the system.

HOL support forward and goal directed proofs. Other styles are available as libraries. In forward proofs, the user begins with previously proved theorems and applies inference rules to reach the desired theorem. Since it requires the exact details of a proof in advance, the forward proof method is often not the easiest solution. A backward or a goal directed proof method is the reverse of the forward proof method. It is based on the concept of a *tactic*; a ML function that breaks goals into simple subgoals. In backward proof method, the user starts with the main goal and uses tactics to decompose it to simpler intermediate subgoals. Some of these intermediate subgoals can be discharged by matching axioms or assumptions or by applying built-in decision procedures. The above steps are repeated for the remaining intermediate goals until no further subgoals are left and hence the desired theorem is proved.

HOL has a rudimentary library facility. Many libraries were supplied by HOL users.

4.1 Formalization of Real Numbers in HOL

The first construction of the reals in a computer theorem prover was by Jutting [9], who, in a pioneering effort, translated the famous *Grundlagen der Analysis* into Automath. All the stream line higher-order-logic theorem provers these days include the formalization of real numbers and some real analysis theories. Harrison [4] formalized real numbers and a significant portion of real analysis in HOL using a new version of Cantor's method [2]. This approach has the merit of being more systematic and keeps the primitive basis of the system small and uncluttered when compared to the formalization of real mathematics in other theorem provers, such as Mizar[1] and PVS [6], which is based on assuming the axioms of real numbers. Harrison's work [4] includes the formalization of topology, limits, sequences and series, differentiation and integration and is part of the current distribution of the HOL system.

4.2 Mutual Recursion in HOL

Slind [7] addressed the act of soundly introducing a recursive definition. He built a pattern-matching style recursive function definition facility, based on mechanically proven wellfounded recursion and induction theorems.

Slind [7] definition of recursive functions is based on the idea of wellfoundeness, a notion from set theory. A mathematical relation is wellfounded if it admits no infinite decreasing chains.

In the context of programs, if the arguments to recursive calls can be placed in a wellfounded relation, the function will terminate. In the context of logic,

[1] http://www.mizar.org/

the function is total.

Functions $f_1 ... f_k$ are mutually recursive when a call to some f_i results in a recursive call to a different f_j, and vice versa.

5 Formalization of the Yee Algorithm

We define the Yee Algorithm using Hol_defn; Hol_defn makes the requested definition, but defers the proof of termination to the user. It is used when termination cannot be automatically proved.

There are two problems to deal with when trying to prove termination. First, we have to understand, intuitively and then mathematically, why the function terminates. Second, we have to phrase this in HOL.

The goal is to find a wellfounded relation on the arguments to the Yee algorithm. There are number of basic and advanced means of specifying well-founded relations. The most common starting point for dealing with termination problems for recursive functions is to find some function, known as a measure under which the arguments of a function call are larger than the arguments to any recursive calls that result.

We prove that ceiling function is strictly monotone and that $ceiling(x-2) < ceiling(x)$.

When given a defn and a quotation denoting a termination relation for a function, the function WF _ REL _ TAC initiates the termination proof.

6 Conclusion

We have presented in this paper the formalization of the Yee algorithm, the first step toward the formalization of FDTD techniques using HOL Theorem Prover.

References

[1] J.P. Berenger, A Perfectly Matched Layer for the Absorption of Electromagnetic Waves, *Journal of Computational Physics,* **114** (1994), 185-200.

[2] D. G. Cantor, Irreducible polynomials with integer coefficients have succinct certificates, *Journal of Algorithms,* **2**, (1981), 385-392.

[3] A. Church, A formulation of the simple theory of types, *Journal of Symbolic Logic,* **5** (1940), 56-68.

[4] J. Harrison, *Theorem Proving with the Real Numbers (Distinguished dissertations),* Springer, 1998.

[5] R. Milner, A theory of type polymorphism in programming, *Journal of Computer and System Sciences,* **17** (1978), 348-375.

[6] S. Owre, N. Shankar, J. M. Rushby and D. W. J. Stringer-Calvert, *PVS System Guide,* Computer Science Laboratory, SRI International, Menlo Park, CA, September 1999.

[7] K. Slind, *Reasoning about Terminating Functional Programs,* PhD Thesis, TU Munich, 1999.

[8] A. Taflove and M. E. Brodwin, Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell's equations, *IEEE Trans. Microwave Theory Tech.,* **23** (1975), 623-630.

[9] L.S. van Benthem Jutting, *Checking Landau's Grundlagen in the AUTOMATH System,* PhD thesis, Eindhoven University, Eindhoven, The Netherlands, 1979.

[10] K. S. Yee, Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media,*IEEE Trans. on Ant. and Prop.,* **14** (1966), 302-307.

Received: January, 2010