01 Jul 2007

An Efficient Implementation of Parallel FDTD

Xiaohe Chen

Michael A. Cracraft

Yaojiang Zhang
Missouri University of Science and Technology, zhangyao@mst.edu

Jianmin Zhang

et. al. For a complete list of authors, see https://scholarsmine.mst.edu/ele_comeng_facwork/863

Recommended Citation

X. Chen et al., "An Efficient Implementation of Parallel FDTD," *Proceedings of the IEEE International Symposium on Electromagnetic Compatibility (2007, Honolulu, HI)*, Institute of Electrical and Electronics Engineers (IEEE), Jul 2007.
The definitive version is available at https://doi.org/10.1109/ISEMC.2007.197

An Efficient Implementation of Parallel FDTD

Xiaohe Chen, Michael Cracraft, Yaojiang Zhang,
Jianmin Zhang, James L. Drewniak
Department of Electrical and Computer Engineering
Univeristy of Missouri-Rolla, Rolla, Missouri 65409
Telephone: (573)341-4262
Fax: (573)341-4532
Email: xchb2@umr.edu

Bruce Archambeault, Samuel Connor
IBM Corp.
3039 Cornwallis Rd
RTP, NC, USA 27709
Telephone: (919)486-0120
Email: barch@us.ibm.com

***Abstract*— A parallel FDTD algorithm has been realized based on 1-D domain decomposition. Data communication among different adjacent processors is manipulated by a self-defined C++ class (MPI_FDTD) in which portable functions of the message passing interface (MPI) library are used. The details of the implementation are discussed. EMC applications of the code such as cross talk of traces, cavity, vias, as well as an IC package are provided to demonstrate the parallel efficiency.**

I. Introduction

The Finite-Difference Time-Domain (FDTD) method is a popular numerical solution for Maxwell equation[1]. It has broad applications in the field of signal integrity and electromagnetics. Although the computational ability has advanced in the recent few decades, the FDTD solution scale is still limited due to the algorithm complexity and memory consumption. Usually, a serial version of FDTD can only simulate up to several million of total number of cells. Simulation of complex or fine structure such as integrated circuits packaging is beyond the capability of a serial FDTD code. Therefore, a parallel FDTD code is required to fully take advantage of the modern computer resources located in supercomputers or PC Clusters[2][3].

This paper presents a 1-D parallel solution to the problem with cluster computation. The 1-D parallel solution entails decomposing the computation domain along one direction (x,y,or z). Each domain is assigned to one processor. Besides the merit of easy implementation, this method requires minimal data communication among different processors because each processor only needs to send or receive data from at most two adjacent processors. Moreover, from the description below, only the boundary fields are required in inter-processor communications. Two major modules are introduced and parallel applications in the electromagnetic field are provided. The performance of parallel FDTD is then discussed.

II. Methodology

FDTD is based on the time domain differential form of Maxwell's equations. Its flexibility in modeling arbitrary geometry and materials makes it easy to divide the whole computational domain into many sub-domains in 3-D space[1].

In this section, parallel computing technique is briefly introduced and the 1-D domain decomposition method is explained.

A. Message Passing Interface (MPI)

The Message Passing Interface (MPI) is a standard distributed compuational function library. There are two major flavors of the library: LAMMPI and MPICH, developed by Indiana Univeristy and Argonne National Lab respectively. The parallel FDTD is implemented to support both libraries.

B. 1-D Domain Decomposition

In the presented FDTD implementation, cells are categorized into three major types: computation domain cells, boundary cells and multi-term Debye material cells. The boundary cells can be treated as special materials; the multi-term Debye material will need special process when being distributed.

Parallel FDTD requires decomposing the computational domain properly so that the resultant sub-domains can be distributed into different processors. There are many domain decomposition approaches. One approach is to split along three dimensions. However, three dimensional splits complicate the process of special components, material interfaces and boundary handling while it does not improve the computational efficiency[2].

Equation 1 shows a boundary iteration function for perfect matched layer (PML) cells.

$$\begin{aligned} H_{xy}^{x+\frac{1}{2}}|_{i,j+\frac{1}{2},k+\frac{1}{2}} &= e^{\rho_z \frac{\Delta t}{\mu}} H_{xy}^{x+\frac{1}{2}}|_{i,j+\frac{1}{2},k+\frac{1}{2}} \\ &+ \frac{1-e^{\rho_z \frac{\Delta t}{\mu}}}{\rho_z \Delta z}(E_y^n|_{i,j+\frac{1}{2},k+1} - E_y^n|_{i,j+\frac{1}{2},k}) \end{aligned} \quad (1)$$

Because the FDTD algorithm is based on first order differential equations, each processor only needs allocate the memory for updating the field vectors in its corresponding sub-domains. To store the communication data from adjacent processors, one more layer of cells is required.

III. Parallel Architectures

A. Architectures

The parallel FDTD algorithm uses three independent modules to interact with each other.

- FDTD main program (algorithm implementation)
- MPI_FDTD control class
- Parallel compatible probe

1-4244-1350-8/07/$25.00 ©2007 IEEE

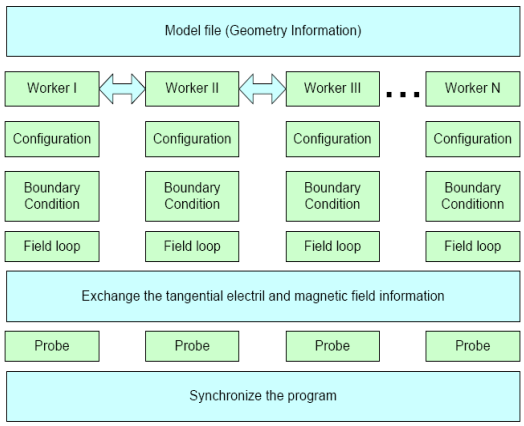

Fig. 1. Parallel FDTD flowchart

The original serial code was modified to add some preliminary data structures to store the information of process boundary, i.e. the memory distance, the loop boundary location and the process boundary buffers.

A stand-alone MPI control class, MPI_FDTD was designed to control all the parallel computing process using the Message Passing Interface (MPI) [4].

The basic parallel algorithm is given below.

1) Initialize the MPI library
2) Calculate, verify the process boundary and distribute the memory allocation
3) Exchange the tangential fields information (H_y, H_z, E_y, E_z) between processors
4) Finalize the MPI functions

The parallel compatible probes assign different processor tags to the probes at different locations. It also handles the movie function to record the field information on a plane in the 3D space.

The FDTD main program interacts with its MPI_FDTD class by calling its member functions. At the beginning of the code, all the MPI functions are initialized. Once the geometry information is known, the process and memory boundary location are calculated. The probes are assigned to their corresponding worker nodes. At the process boundary, the workers will allocate one more layer of cells as the a transfer buffer. The electromagnetic field information on the boundary layer of two adjacent processors are stored in the buffer for the next time step iteration.

After setting up sub-domains, each processor needs to check for any special components, such as thin wires, thin slots, current sources, current probes, and pseudo-wires, across boundary layers. If these components are distributed on different processors, special processing and data exchange among processors are required. This makes the implementation very cumbersome without additional benefits. Therefore, in our code, the MPI_FDTD will shift the boundary layer automatically to avoid such components across the boundary.

B. Field Data Exchange

There are a number of parallel architectures, such as single-instruction multi-data (SIMD), multiple-instruction multiple data (MIMD), and single-program multiple-data (SPMD). The parallel FDTD implements a non-supervised SIMD structure. [5] Each processor in the cluster uses the same algorithm to calculate the neighbor locations and the length of the process boundary based on its own tags. The workers also transfer the data and communicate with each other directly. This architecture has three advantages: it does not require a master to coordinate and dispatch the work load; it simplifies the coding structure; and it reduces unnecessary communications.

Each processor in the cluster needs to exchange the field information at the boundary layer. As shown in Fig. 2, the tangential fields of the boundary layer (H_y, H_z, E_y, E_z) are exchanged and stored in the boundary buffers.

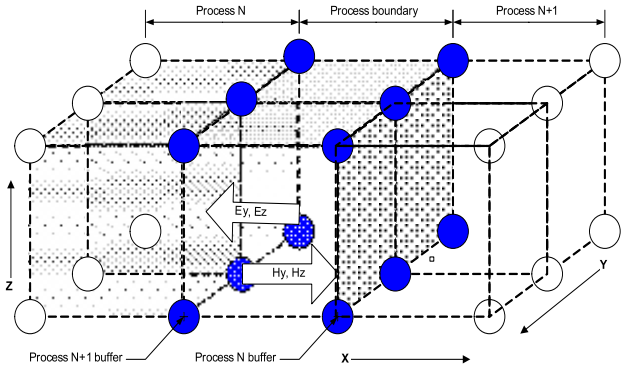

Fig. 2. Parallel FDTD field transfer directions

To facilitate the following description, let *West* be the negative *X* direction and let *East* to be the postive *X* direction. As shown in Figure 3, the updating of the magnetic field is only depending on the electric field of its own cell and the succeeding cells ($i+m$, $j+m$, $k+m$, $m = 0$ or 1). Similarly, the updating of electric fields needs the magnetic fields on itself and the preceding cells ($i-m$, $j-m$, $k-m$, $m = 0$ or 1). Therefore, The parallel FDTD program exchanges the field information by transfering magnetic field eastbound and electric field westbound.

C. Memory Distribution

A single PC does not have enough memory for large scale problems. Memory distribution is one of the motivation for parallel computing to fully utilize the computer resources. The class of modelling problems considered typically exceeds 4GB. The FDTD program uses computational domain structure and a lookup material property table to assign the material information to the cells. The process boundary distance is

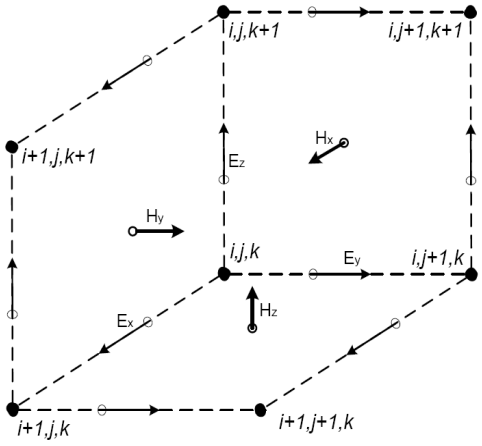

Fig. 3. Parallel FDTD field dependency

calculated for the computer to allocate the memory. A local counter is set up for each processor to calculate the size of the local material property table needed.

D. Dispersive Materials

Debye-type dispersive materials have frequency-dependent permeability and permittivity which can be characterized by the following complex functions

$$\varepsilon_r(\omega) = \varepsilon_\infty + \sum_{p=1}^{P} \frac{\triangle\varepsilon_p}{1 + j\omega\tau_p^e} \quad (2)$$

$$\mu_r(\omega) = \mu_\infty + \sum_{p=1}^{P} \frac{\triangle\mu_p}{1 + j\omega\tau_p^h} \quad (3)$$

where p denotes the number of the Debye poles. $\triangle\varepsilon_p = \varepsilon_{sp} - \varepsilon_\infty$, $\triangle\mu_p = \mu_{sp} - \mu_\infty$, and ε_{sp} and ε_∞ are the static and optical relative permittivity respectively. $\tau_p^{(e,h)}$ is the relaxation time associated with pth pole.

Using time domain Auxiliary Differential Equations (ADE-FDTD) (add reference) to solve the Debye material has been proved to be a stable approach for dispersive dielectric and magnetic materials. Unlike the normal Maxwell's equation solutions, the ADE method introduces auxiliary electrical and magnetic polarization sources $\mathbf{J}_p$ and $\mathbf{M}_p$. Its magnetic field time marching equation can be described as Equation. 4.

$$\mathbf{H}^{n+\frac{1}{2}} = \left(\frac{2\mu_0\mu_e - \sigma_h\Delta t}{2\mu_0\mu_e + \sigma_h\Delta t}\right)\mathbf{H}^{n-\frac{1}{2}} + \left(\frac{2\Delta t}{2\mu_0\mu_e + \sigma_h\Delta t}\right) \cdot \left[-\nabla\times\mathbf{E}^n - \frac{1}{2}\sum_{p=1}^{P}\left(1 + k_p^h\right)\mathbf{M}_p^{n-\frac{1}{2}}\right] \quad (4)$$

where the effective permittivity μ_e is given by,

$$\mu_e = 2\mu_0(\mu_\infty + \sum_{p=1}^{P}\beta_p^h) \quad (5)$$

with,

$$\beta_p^h = \frac{\Delta\mu_p\Delta t}{2\tau_p^h + \Delta t} \quad (6)$$

$$k_p^h = \frac{2\tau_p^h - \Delta t}{2\tau_p^h + \Delta t} \quad (7)$$

The magnetic current polarization source is updated by Equation. 8

$$\mathbf{M}_p^{n+\frac{1}{2}} = k_p^h\mathbf{M}_p^{n-\frac{1}{2}} + 2\mu_0\beta_p^h\left(\frac{\mathbf{H}^{n+\frac{1}{2}} - \mathbf{H}^{n-\frac{1}{2}}}{\Delta t}\right) \quad (8)$$

Since the polarization sources are only depending on the local Debye material properties and the rate of the field change $\frac{\partial H}{\partial t}$, there should be no need for extra information exchange except the normal field information in four tangential directions.

However, when dealing with an interface between different materials, different processes might exchange material informations up to two layers of cells to calculate the effective permittivity and permeability along the process boundary.

IV. Performance and Applications

The parallel FDTD has been used on several large scale electromagnetic problems including an IC package simulation, a finite grounded microstrip problem.

A. Performance and Efficiency

A metric equation has been derived to evaluate the performance of the parallel FDTD programm on clusters.

$$\gamma = \frac{N}{t\cdot m}\cdot C \quad (9)$$

where *t* is the total execution time in hours, *N* is the total time steps executed, *C* is the total number of the cells in millions, *m* is the total processors used in the simulation.

Table I lists a number of models tested and its performance.

B. Finite Grounded microstrip on a PCB

One of the models computated by parallel FDTD is to validate the German Benchmark #12, the finite grounded microstrip interconnect on a PCB board.

As shown in Fig. 4, the microstrip is grounded by a 30 cm $\times$ 50 cm ground plane. A 50 Ω voltage source is added to the left side of the 30 cm microstrip. A 50 Ω is added to the other side of the trace. The source is configured as a periodical trapezoidal voltage waveform with 10%90% rise time and fall time equal to 1.4 ns, and period of 30 ns.

The model for this problem consumes 56.33 million cells and takes about 20 hours to finish.

A S21 result compared with other EM tools including CST, HFSS and ADS is shown in Fig. 5.

TABLE I

PARALLEL FDTD BENCHMARK METRIC

FDTD Model	Total cells [M]	Memory [GB]	Processors	Performance ($gamma$)
Cross talk trace model 2,3,6,7	41.25	6.60	32	2,837
Fine model for DAT_02	60.06	9.09	32	2,432
Model for DAT_06	41.17	6.43	32	2,755
Model for DAT_01	38.93	5.61	32	2,937
Finite grounded microstrip*	56.33	2.41	32	3,391
Cavity model	0.913	0.067	6	6,521
Via model	2.354	0.152	6	6,277

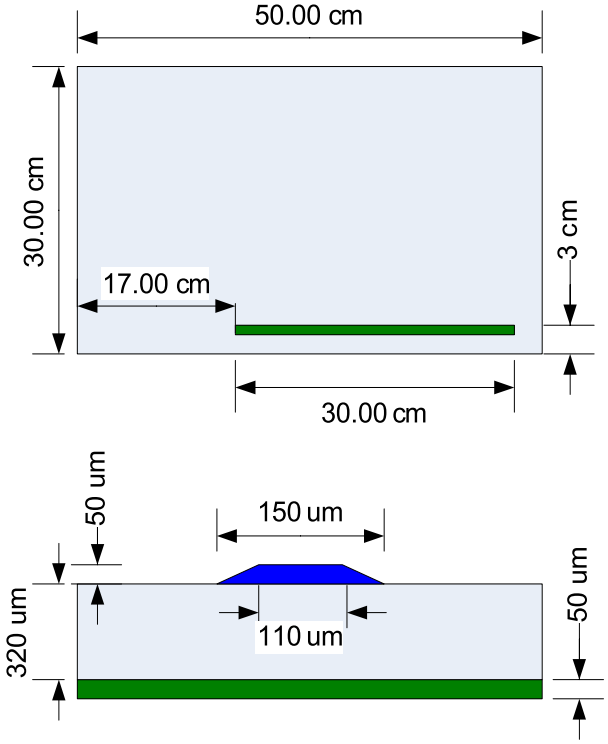

Fig. 4. Finite grounded microstrip

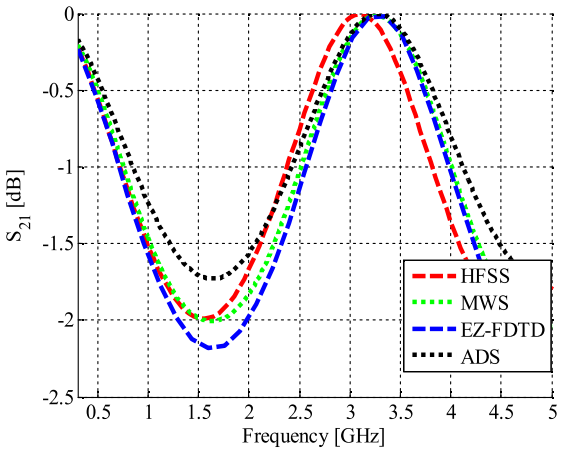

Fig. 5. Finite grounded microstrip S21 results

C. IC BGA packaging bounding trace cross talk simulation

An IC BGA packaging problem is shown in the Fig. 7. The package size is 30 mm x 30 mm, and the total number of conductive layers from the chip side to the BGA side is eight. Thin film dielectric is used as the substrates. The challenge for the simulation is that the geometry is very complex, and the gridding needs to reflect the fine structure. The total number of the cells for even a single trace may exceed the capability of a serial FDTD program. The region of interest contains 18 data traces from chip side to BGA solder points and one pair of differential clock traces. The traces propagate mainly on the FC3 layer, reaching the bottom layer through a number of vias. The typical trace width is from 20 μm to 50 μm, and the thickness of the copper layers is approximately 15 μm.

The model file contains eight copper layers, approximately 1400 vias, and a signal trace (DAT_06). The total number of cells is 41.7 million and the approximate execution time is about 20 hours.

The FDTD geometry is able to accurately simulate the transmission line delay on a single trace as shown in Table II. A cross talk simulation is also presented in Fig. 7, The model includes four bounding trace, DAT_02, 03, 06, 07. A 50 Ω voltage resistive source is added to the chip side of the DAT_06 as the agressor, All other three traces are terminated with 50 Ω on the chip side and open ended on the BGA side. The the simulation is able to reflect the amplitude and time delay (around $2t_d$) of the cross talk. A further investigation on how to model the other measurement effects on the package is still needed.

TABLE II

STRIP LINE DELAY SIMULATION

TRACE	Measured t_d	Simulated t_d	Calculated t_d
DAT_02	118.6 ps	107.5 ps	112.0 ps
DAT_06	119.3 ps	117.0 ps	117.0 ps

V. CONCLUSION

A stable, highly efficient parallel method of FDTD architecture is presented. MPI_FDTD control module is applied to

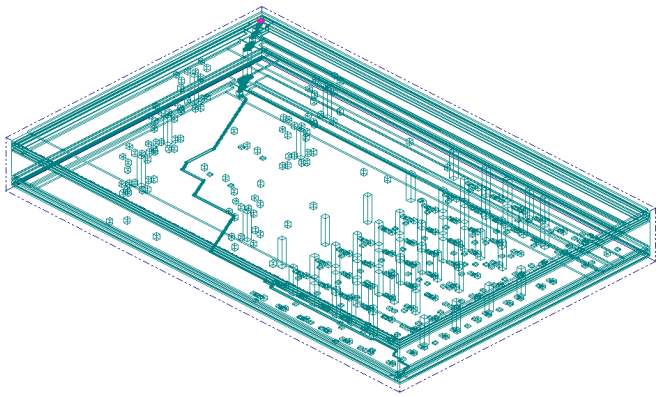

Fig. 6. FDTD model for a single trace in an IC package

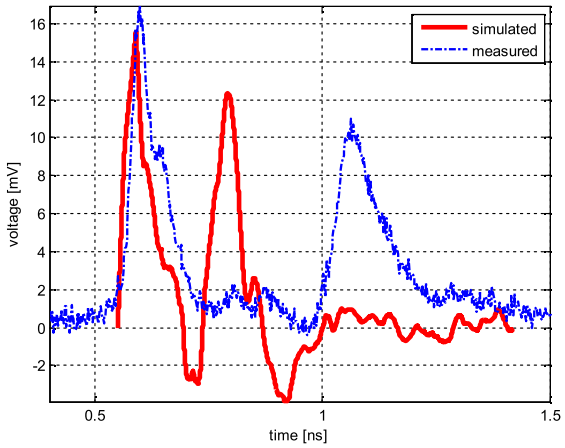

Fig. 7. Crosstalk measurement and simulation

the FDTD algorithm to distribute the memory and computation sub-domains into different processors. The speed-up efficiency and performance of the parallel FDTD program are analyzed. Several applications in EMC analysis are provided to demonstrate the efficiency of the code.

REFERENCES

[1] Allen Taflove, Susan C. Hagness, *Computational Electrodynamics - The Finite-Difference Time-Domain Method*, 3rd ed. Artech House, Norwood, MA 02062, 2005.

[2] Michael Cracraft, *Parallelizing a Finite-Difference Time-Domain code using the Message Passing Interface*, UMR Master Thesis, Electrical Engineering, 2002.

[3] W. H. Yu, Y. J. Liu, T. Su, N.-T. Huang, and R. Mittra, "A robust parallel conformal finite-difference time-domain processing package using the MPI library," *IEEE Antennas & Propagat. Mag.*, vol. 47, no. 3, pp. 39-59, 2005.

[4] Gropp W. E. Lusk, A. Skjellum, *Using MPI*, 3rd ed. Cambridge, MA, MIT Press, 1999.

[5] Cameron Hughes, Tracey Hughes, *Parallel and Distributed Programming Using C++*. Boston, MA, Addison Wesley, 2003.

[6] Xiaohe Chen, James L. Drewniak, Jianmin Zhang, Michael Cracraft, Bruce Archambeault, Sameul Connor, *Large Scale Signal and Interconnect FDTD Modeling for BGA package*, IEEE 15th Topical Meeting on Electrical Performance of Electronic Packaging Oct 23-25, 2006.

[7] OkoniewskiM., M. Mrozowski, and M. A. Stuchly, "Simple treatment of multi-term dispersion in FDTD," *IEEE Microwave Guided Wave Lett.*, vol.7, pp. 121-123, 1997.