

Implementation of Three-Dimensional FPGA-Based FDTD Solvers: An Architectural Overview

James P. Durbano
EM Photonics, Inc.
durbano@emphotonics.com

Fernando E. Ortiz
John R. Humphrey
Dennis W. Prather
University of Delaware
{ortiz, humphrey, dprather}@ee.udel.edu

Mark S. Mirotznik
The Catholic
University of America
mirotznik@cua.edu

No longer relegated to radio frequency (RF) engineers, antenna designers, and military applications, electromagnetic analysis has become a key factor in many areas of advanced technology. From 3 GHz PCs and wireless computer networks, to PDAs with Internet capabilities and the seemingly ubiquitous cell phone, it seems that electronic designs increasingly require electromagnetic characterization. To facilitate such analysis, numerical techniques have been developed that allow computers to easily solve Maxwell's equations.

Maxwell's equations, which govern electromagnetic propagation, are a system of coupled, differential equations. As such, they can be represented in difference form, thus allowing their numerical solution. By implementing both the temporal and spatial derivatives of Maxwell's equations in difference form, we arrive at one of the most common computational electromagnetic algorithms, the Finite-Difference Time-Domain (FDTD) method [1]. In this technique, the region of interest is sampled to generate a grid of points, hereafter referred to as a mesh. The discretized form of Maxwell's equations is then solved at each point in the mesh to determine the associated electromagnetic fields.

Although FDTD methods are accurate and well defined, current computer-system technology limits the speed at which these operations can be performed. Run times on the order of hours, weeks, months, or longer are common when solving problems of realistic size. Some problems are even too large to be effectively solved due to practical time and memory constraints. The slow nature of the algorithm primarily results from the nested for-loops that are required to iterate over the three spatial dimensions and time.

To shorten the computational time, people acquire faster computers, lease time on supercomputers, or build clusters of computers to gain a parallel processing speedup [2], [3]. These solutions can be prohibitively expensive and frequently impractical. As a result, an approach that increases the speed of the FDTD method in a relatively inexpensive and practical way is required. To this end, people have suggested that an FDTD accelerator, i.e., special-purpose hardware that implements the FDTD method, be used to speed up the computations [4]-[8]. However, none have succeeded in developing a practical implementation, nor a full three-dimensional solver.

In this extended abstract, we present an architecture that overcomes the previous limitations. We begin with a high-level description of the computational flow of this architecture.

The computational datapath begins with the Counting and Control Unit (CCU). In addition to containing global system

data, the CCU produces the coordinates and type (electric or magnetic) of the next field to be computed. These coordinates are then passed to the Data Dependence Unit (DDU). The DDU is responsible for determining all values necessary to update the individual field components at this node (i.e., which surrounding field values are required).

The coordinates output by the DDU are then passed into a RAM Address Decoder (RAD). This unit takes a given field component (e.g., $E_x(i,j,k)$) and determines its location in memory. By including multiple DDU and RAD units in the design, several field components can be updated simultaneously. Because this generates numerous read requests, the Memory Switching Unit (MSU) was developed to coordinate all memory transactions.

The majority of the problem data are stored in three RAM banks. Each RAM contains x , y , or z -directed fields and the material type of each node (e.g., air, water, silicon). As data are fetched from RAM, they are stored in register banks until all necessary data have been retrieved and the system is ready to update the field.

Before the field-update computation occurs, however, several material coefficients must be determined. These coefficients are used in the computation of the field-update equation and take into account the material properties of the medium (e.g., permittivity, permeability, conductivity). To determine the coefficients, the material types are passed to the Material Lookup Table (MLUT). The MLUT reads in bit vectors representing a given material and returns the various coefficients corresponding to those materials.

The surrounding field values and material coefficients must then be routed to the appropriate Computation Engine (CE), which updates the given field component based on the discretized forms of Maxwell's equations. Several CEs are included in the design, allowing the system to update multiple field components in parallel. The updated values are then passed back to the MSU for storage in RAM.

In order to test our architectural ideas, a prototyping board with a Xilinx Virtex-II 6000 FPGA, several RAM banks, and a PCI interface was acquired. The user describes the design to analyze by means of a CAD front end developed by EM Photonics, Inc. The front-end software then sends the appropriate data, such as the mesh size and the number of timesteps to execute, to the hardware via the PCI bus. The FDTD accelerator proceeds to update the fields, periodically sending the results back to the host computer for post-processing and visualization.

The benchmark problem was an air-filled cavity surrounded by perfect electric conductor (PEC) walls. The cavity was excited by a z-directed, sinusoidal point source (of unity amplitude) located at the center of the resonator. The simulation was run for 5,000 timesteps with a mesh size of 43x43x43. In order to analyze the error, a point detector was placed in the corner of the cavity.

The hardware results were then compared with the results obtained from C and MATLAB 6.1 programs solving the same problem on 1.13 and 2.0 GHz PCs. A C implementation was chosen to perform the speed analysis, whereas a MATLAB implementation was chosen for error analysis. This allowed us to optimize the C program for speed and the MATLAB program for error measurements.

The average absolute error was on the order of 10^{-7} with the average percentage error around 0.13%. This numerical error is a result of two primary factors. First, MATLAB is a double-precision (64 bit) language whereas the hardware implementation supports only single-precision (32 bit) arithmetic units. If precision is of the utmost importance, however, double-precision arithmetic units can easily be implemented. The second factor that contributes to the error is the computation of the source field. For simplicity, the sine function was implemented as a lookup table (LUT) with only 16K entries. In future implementations, more entries will be included in the LUT to increase resolution or other techniques, such as the CORDIC algorithm, will be used to generate the source [9].

In terms of processing power, the 14 MHz hardware had an average throughput of approximately 150,000 nodes per second (150 Knps¹). This is 5.66 times slower than the processing power of C running on a 1.13 GHz PC (849 Knps) and 8.55 times slower than C on a 2.0 GHz machine (1,282 Knps). *Note that although the PC is clocked over 142 times faster than the hardware, the hardware is less than 9 times slower.*

Certainly a design that is slower than existing solutions is not desired! However, this was a proof-of-concept design that served not only to implement our basic architectural ideas, but also to achieve the first three-dimensional FDTD accelerator implementation in physical hardware. As such, these results were obtained on a preliminary, non-optimized design. A detailed analysis indicates that overlapping the computations of different nodes will result in a threefold increase in speed. Also, because the throughputs of our accelerator increase linearly with clock frequency, by increasing the clock frequency from 14 MHz to 100 MHz, a common FPGA system speed, a sevenfold increase in throughput is possible. Although the current design is almost nine times slower than a 2.0 GHz PC running optimized C code, after the above design modifications are made, the hardware throughput will be almost two and a half times that of a 2.0 GHz PC. Note that modifying the design to work at 100 MHz is not an unreasonable goal, as the most complex units in the design are the floating-point arithmetic units, which are already capable of speeds in excess of 100 MHz. The overall clock frequency had to be reduced for some non-optimized units related to the routing of data. These units can be pipelined, thus permitting increased clock frequencies.

¹ Knps = thousands of nodes processed per second, where the time to process a node is the time to update all of the fields at that node.

Finally, it should be noted that these results are from a commercial, off-the-shelf prototyping board. Because the design had to be mapped into this general-purpose board, several architectural optimizations (such as increased parallelism) could not be implemented. Initial calculations indicate that a customized board can provide speed increases of *at least two orders of magnitude* through the addition of multiple RAM banks, increased clock frequencies, and a more efficient use of memory.

To the best of our knowledge, this work represents the first successful three-dimensional FDTD algorithm in hardware. We are currently working on an optimized version of this architecture and a custom printed circuit board to support our design. This will provide increased computational speeds, which we predict will easily surpass desktop computers, and will ultimately rival the performance of computer clusters.

References

- [1] K. S. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, vol. 14, pp. 302-307, 1966.
- [2] H. Jordan, S. Bokhari, S. Staker, J. Sauer, M. ElHelbawy, and M. Picket-May, "Experience with ADI-FDTD techniques on the Cray MTA supercomputer," in *Proc. of the SPIE - Commercial Applications for High-Performance Computing*, vol. 4528, pp. 68-76, 2001.
- [3] G. A. Schiavone, I. Codreanu, R. Palaniappan, and P. Wahid, "FDTD speedups obtained in distributed computing on a Linux workstation cluster," *IEEE Antennas and Propagation Society, AP-S International Symposium (Digest)*, vol. 3, pp. 1336-1339, 2000.
- [4] J. R. Marek, M. A. Mehalic, and J. Andrew J. Terzuoli, "A dedicated VLSI architecture for Finite-Difference Time Domain calculations," in *Proc. of The 8th Annual Review of Progress in Applied Computational Electromagnetics*, Naval Postgraduate School, Monterey, CA, 1992.
- [5] R. N. Schneider, L. E. Turner, and M. M. Okoniewski, "Application of FPGA technology to accelerate the Finite-Difference Time-Domain (FDTD) method," in *Proc. of The Tenth ACM International Symposium on Field-Programmable Gate Arrays*, Monterey, CA, 2002.
- [6] P. Placidi, L. Verducci, G. Matrella, L. Roselli, and P. Ciampolini, "A custom VLSI architecture for the solution of FDTD equations," *IEICE Transactions on Electronics*, vol. E85-C, pp. 572-577, 2002.
- [7] L. Verducci, P. Placidi, G. Matrella, L. Roselli, F. Alimenti, P. Ciampolini, and A. Scorzoni, "A feasibility study about a custom hardware implementation of the FDTD algorithm," in *Proc. of The 27th General Assembly of the URSI*, Maastricht, Netherlands, 2002.
- [8] J. P. Durbano, "Hardware implementation of a 1-dimensional Finite-Difference Time-Domain algorithm for the analysis of electromagnetic propagation," M.E.E. Thesis, Department of Electrical and Computer Engineering, University of Delaware, Newark, USA, 2002.
- [9] R. Andracka, "A survey of CORDIC algorithms for FPGA based computers," in *Proc. of The ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, Monterey, CA, USA, 1998.