Finite-Difference Time-Domain Method in Custom Hardware?

Ryan N. Schneider, *Member, IEEE*, Michal M. Okoniewski, *Senior Member, IEEE*, and Laurence E. Turner, *Member, IEEE*

***Abstract*—While the finite-difference time-domain (FDTD) method is very successful in electromagnetics, it is computationally intensive. Reducing the runtime of these simulations, by an order of magnitude or more, would greatly increase the productivity of FDTD users and open new avenues of research. A dedicated hardware implementation that accelerates FDTD computations could provide a means to attain that goal.**

As the first step, we have implemented a one- and two-dimensional FDTD method in hardware. The experiment proved that computational speed can be increased by as much as two orders of magnitude, and is independent of the number of cells in the simulation.

***Index Terms*—Acceleration, digital circuits, finite-difference time-domain (FDTD) methods, field programmable gate arrays.**

I. Introduction

THE FINITE-DIFFERENCE time-domain (FDTD) method [1] has been successfully and very widely applied to many electromagnetic problems [2]. The algorithm is, however, computationally intensive, as it can easily involve upwards of millions of computational cells for each iteration. The past decade has seen a large increase in computational resources at declining costs, but simulations can still run for several days on multiprocessor supercomputers. Decreasing the run time of this algorithm would greatly benefit FDTD users and open up new areas of research.

The objective of this research is to accelerate the computation of the FDTD algorithm by a factor of 10–100 times, by using custom, dedicated hardware, integer arithmetic, and fine-grained parallelism. The long-term goal is to integrate this level of acceleration into existing FDTD software platforms.

The acceleration is achieved by simultaneously applying a number of strategies: 1) Calculations are performed on custom, field-programmable gate-array (FPGA) based hardware. Hardware based acceleration was previously attempted by Marek *et al.* [3]. They presented a simulated, hardware description language (HDL) coprocessor in a Sparc2 system, with a predicted acceleration on the order of five to nine times. 2) Hardware computations are performed using integer arithmetic. Nearly all published implementations of FDTD use floating-point calculations, except Grinin [4], where integer FDTD code is used to avoid the expense of floating point calculations on a 16-bit integer-optimized processor. 3) Inherent parallelism of FDTD is fully exploited in hardware.

Manuscript received April 12, 2002; revised June 27, 2002. This work was supported in part by the National Science and Engineering Research Council (NSERC) of Canada. The review of this letter was arranged by Associate Editor Dr. Shigeo Kawasaki.

The authors are with the University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: ryan.schneider@ieee.org; michal@enel.ucalgary.ca; turner@enel.ucalgary.ca).

Digital Object Identifier 10.1109/LMWC.2002.805948

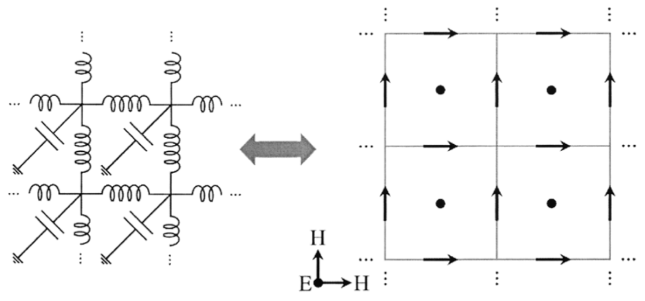


Fig. 1. The 2-D FDTD grid and inductor–capacitor equivalent.

II. Implementation

For simplicity, a one-dimensional (1-D) FDTD cell is discussed in this section, although both 1-D and two-dimensional (2-D) cases have been implemented in hardware and verified.

A. Inductor–Capacitor Representation of FDTD

Gwarek [5] provides a representation of the 2-D FDTD algorithm in terms of inductors and capacitors (Fig. 1).

The inductor–capacitor structure is similar to a passive, infinite LC ladder filter. Thus, traditional digital filter implementation techniques can be used to implement an FDTD calculation using digital hardware.

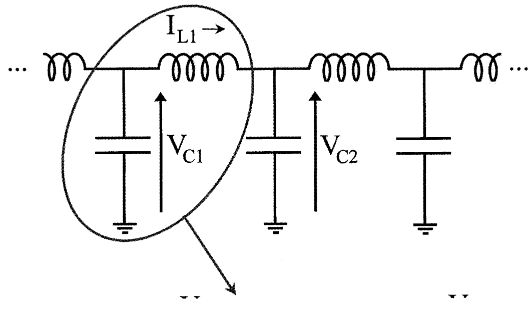
B. 1-D FDTD Cell

The 1-D FDTD can be represented as an unterminated LC ladder network (a special case of Fig. 1), and is further depicted in Fig. 2.

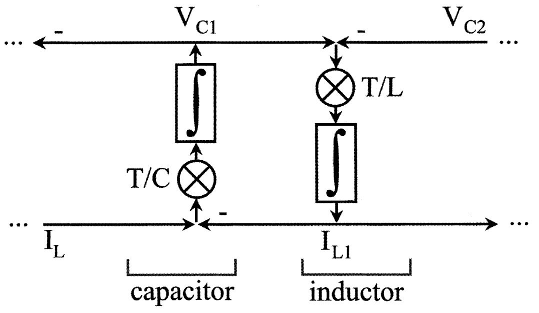
Each integrator in the voltage–current signal flow graph (Fig. 2) is replaced by a discrete time approximation, denoted as a “lossless discrete integrator” (LDI) [6], shown in Fig. 3.

LDIs implement a centered-difference (trapezoidal) integration algorithm. There is a direct correlation between the traditional FDTD update equations and the analytical evaluation of this inductor–capacitor network using LDIs.

Using signal flow graph manipulation, the delays are rearranged to give the following signal flow graph presented in Fig. 3, which yields itself to a straightforward hardware implementation, shown in Fig. 4.



(a)



(b)

Fig. 2. (a) The 1-D inductor–capacitor FDTD equivalent. (b) Signal flow graph. Voltages are represented by the signals at the top of the signal flow graph, while currents are represented on the bottom.

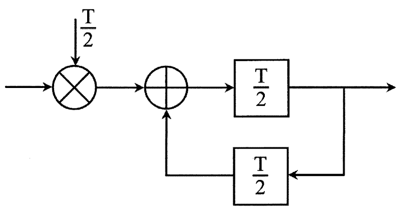


Fig. 3. Lossless discrete integrator.

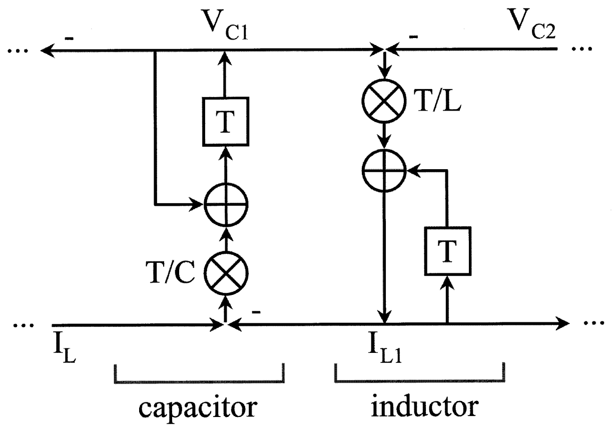


Fig. 4. Signal flow graph of a 1-D FDTD cell.

C. Hardware Implementation

Pipelined, bit-serial arithmetic was chosen in order to implement the signal flow graph of Fig. 4. Although such an approach is considerably slower than a more traditional (in digital computation) bit-parallel implementation, it results in a significantly reduced number of required gates, simplified routing (local connectivity), and short routing lengths.

Integer arithmetic was chosen in an attempt to further reduce the hardware cost and increase the computational speed of the implementation. These gains are offset by the need for larger integer registers to represent the equivalent floating point capacity of traditional FDTD implementations. Full details of the bit-serial hardware implementation are given in a paper the authors have written for the FPGA community [7].

D. Fixed Precision Implications

It was found, experimentally, that using 8-bit coefficients to represent the T/C or T/L multiplier values can lead to unstable simulations. Increasing the coefficient precision to 12-bits provides stability. It is possible that the value of Δt, the time sample interval, could be adjusted such that the coefficients are as close to a convenient integer representation as possible. Once a final implementation has been selected, more research is required in this area.

TABLE I
PERFORMANCE OF SOFTWARE AND HARDWARE FDTD IMPLEMENTATIONS

	$f_{theoretical}$ (GHz)	$f_{calculated}$ (GHz)	% error	run time (ms)
Hardware - 1D	1.4990	1.5196	1.37%	8.49
	2.9980	3.0393	1.38%	
Software** - 1D	1.4990	1.4946	-0.29%	71
	2.9980	2.9970	-0.03%	
Hardware - 2D	1.6655	1.6601	-0.32%	12.5
	4.9966	4.8543	-2.85%	
Software** - 2D	1.6655	1.6547	-0.65%	230
	4.9966	4.8307	-3.32%	

III. Results

A. 1-D and 2-D Resonators

A 1-D, free-space cavity resonator, ten FDTD cells in length and terminated in perfect electric conductors, was implemented as the first test. Although such a resonator represents a trivial example, it is very useful for verification of the algorithm implementation. Errors due to incorrect calculations quickly accumulate and the output becomes either unbounded, or erroneous. Resonant frequencies are several orders of magnitude above the noise floor and narrowband. The coefficients directly relate to the location of these frequencies, further verifying the multiplier structure.

The second, 2-D resonator was 2 cells $\times$ 5 cells. It was terminated on three sides by magnetic walls and the final boundary (on a long side) was terminated in an electric wall. The dimensions of this resonator were small due to the limited size of the hardware device.

The target hardware device is an older generation Xilinx Virtex Family FPGA, XCV300, and it offers 3072 slices. A "slice" is a measure of an atomic unit of hardware resources on an FPGA. Most importantly, Xilinx Virtex slices contain two flip–flops and two four-input lookup tables.

B. Simulation Results

A comparison of the computed versus analytical resonant frequencies is included in Table I on the following page.

Although in some cases the hardware computation (2-D) performs better than a software implementation, in general, we would expect less accuracy due to the limited precision of coefficients, which are expressed as ratios of integers.

C. Computation Speed and Hardware Requirements

The maximum operating frequency of the hardware, for the 1-D case reported by the Xilinx FPGA CAD tools, is 37.7 MHz. For a 32-bit system word length (SWL), a new result is computed every 849 ns (1.18 MHz). Assuming that the observation data could be output from the FPGA at this rate, 10 000 time steps could be computed in 8.49 ms versus 71 ms in software. A 1-D FDTD computational cell occupies 86.5 slices. The 10 cell resonator used 30% of the device or 917 slices. 52 slices are used for data collection leaving 865 slices for the FDTD cells and control structure.

Each 2-D FDTD cell requires 120 Virtex slices. Operating at a serial clock of 32 MHz and with a longer, 40-bit SWL, new results are available every 1.25 μs (fop $=$ 0.8 MHz, 10 000 iterations $=$ 12.5 ms versus 230 ms for a software implementation). It is predicted that the three-dimensional (3-D) computational cell would require 265 slices to represent six fields.

IV. Discussion

The 1-D and 2-D FDTD algorithms have been successfully implemented in hardware. The computation speed is extremely fast and not related to the number of cells. This approach represents maximum possible parallelism because every computational cell of the simulation is implemented on hardware. A larger simulation, with more cells, would require more hardware. It is not feasible, however, to simply extend the current approach to the 3-D case—current FPGA parts are not large enough. For a 3-D case, other approaches are, therefore, explored that build upon the research reported here in a structure that reuses a smaller computational engine to compute larger structures. The current method is, however, already applicable to many 2-D problems, e.g., with rotational symmetry.

References

[1] K. S. Yee, “Numerical solution of initial boundary value problems solving Maxwell’s equations in isotropic media,” *IEEE Trans. Antennas Propagat.*, vol. 14, pp. 302–307, 1966.

[2] A. Taflove, *Advances in Computational Electrodynamics—The Finite Difference Time Domain Method*. Norwood, MA: Artech House, 1998.

[3] J. R. Marek, M. A. Mehalic, and A. J. Terzuoli, “A dedicated VLSI architecture for finite-difference time domain calculations,” *Proc. 8th Annu. Rev. Progress Appl. Comput. Electromagn.*, vol. 1, pp. 546–553, Mar. 1992.

[4] S. V. Grinin, “Integer-arithmetic FDTD codes for computer simulation of internal, near and far electromagnetic fields scattered by three-dimensional conductive complicated form bodies,” *Comput. Phys. Commun.*, vol. 102, no. 1–3, pp. 109–131, May 1997.

[5] W. K. Gwarek, “Analysis of arbitrarily shaped two-dimensional microwave circuits by finite-difference time-domain method,” *IEEE Trans. Microwave Theory Tech.*, vol. 36, pp. 738–744, Aug..

[6] L. T. Bruton, “Low sensitivity digital ladder filters,” *IEEE Trans. Circuits Syst.*, vol. CAS-22, pp. 168–176, Mar. 1975.

[7] R. N. Schneider, L. E. Turner, and M. M. Okoniewski, “Application of FPGA technology to accelerate the finite-difference time-domain (FDTD) method,” *Proc. 10th Int. Symp. Field Programmable Gate Arrays*, Feb. 24–26, 2002.