Custom Hardware Implementation of the Finite-Difference Time-Domain (FDTD) Method

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

University of Calgary, Electrical and Computer Engineering, 2500 University Drive NW, Calgary, Alberta, T2N 1N4, CANADA

***Abstract* — Finite-Difference Time-Domain (FDTD) Analysis is a very popular method for solving electromagnetic problems. The algorithm is computationally intensive and simulations can take several days to run on traditional, multiprocessor supercomputer platforms. 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 hardware implementation of a one-dimensional FDTD computational cell is presented, with the goal of accelerating three-dimensional computations by a factor of 10-100 times. A free-space, cavity resonator is used to successfully verify the FDTD simulation on hardware. Computational speed is very promising and is independent of the number of cells in the simulation. Larger simulations require more hardware. A typical simulation size (100x100x100) is hardware prohibitive, so future work will investigate hardware sharing methods.

I. INTRODUCTION

The Finite-Difference Time-Domain (FDTD) [1] has been successfully and very widely applied to the solution of complex electromagnetic problems [2]. The algorithm is computationally intensive and involves a three-dimensional simulation volume of upwards of millions of computational cells at a time. 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.

This research is novel for a number of reasons. First, initial projections indicate that the FDTD algorithm could be successfully accelerated by an order of magnitude or more. Second, 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) co-processor in a Sparc2 system, with a predicted acceleration on the order of five to nine times. Third, 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.

A. Benchmark

A three-dimensional simulation volume is spatially sampled such that there are at least 10-20 samples per wavelength at the highest frequency of interest. The simulation is run for three to four periods at the lowest frequency of interest, which typically equates to 10,000 time steps. Thus, a typical simulation would have the following properties.

Table 1: Typical Simulation Run-Time

Typical Simulation Size	100x100x100 cells	1.00.E+06	cells
	6 fields per cell	6.00.E+06	fields
Typical Simulation Length	10,000 iterations	6.00.E+10	updates
	8 flops per update equation	4.80.E+11	flops
Estimated run-time	1 Gigaflops per second processor (single CPU)	480	seconds

An assumption is made that a floating-point addition or multiplication takes one flop (floating-point operation). This calculation yields an overly optimistic runtime. It ignores variable memory access speed (cache misses, hard drive paging), operating system overhead, time-sharing and many other problems that exist in traditional computer systems. Finally, it focuses only on the core update equations which update the magnetic and electric fields in three-dimensions for each time step. More advanced

FDTD code would perform additional calculations for sub-cellular structures, complex media and absorbing boundary conditions. Accurate absorbing boundary conditions can add as many as eight layers to the boundaries of the simulation region, which increases the computational requirements by as much as 70%.

II. Implementation

For simplicity, a one-dimensional FDTD cell is discussed. Both one and two-dimensional cases are implemented and verified. Results are also extrapolated to a three-dimensional case. It should be noted that a three-dimensional hardware solution has not been implemented at this time. This is discussed in detail in later sections.

A. Inductor-Capacitor Representation of FDTD

Gwarek [5] provides a representation of the two-dimensional FDTD algorithm in terms of inductors and capacitors. The electric and magnetic fields are represented by voltage in the capacitors and current in the inductors, respectively. This relationship is further demonstrated in Figure 1.

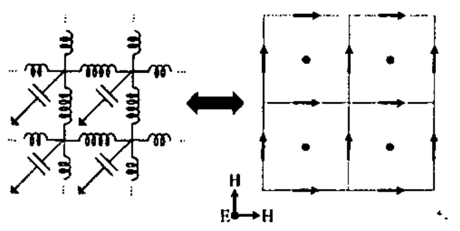

Figure 1: Two-Dimensional FDTD Grid and Inductor-Capacitor Equivalent

The inductor-capacitor structure is very similar to the digital ladder filter structure introduced by Bruton [6]. Thus, traditional digital filter implementation techniques can be used to implement an FDTD calculation using digital hardware.

B. One-Dimensional FDTD Cell

Using Gwarek's work, the one-dimensional representation is just a special case of Figure 1 and is further depicted in Figure 2.

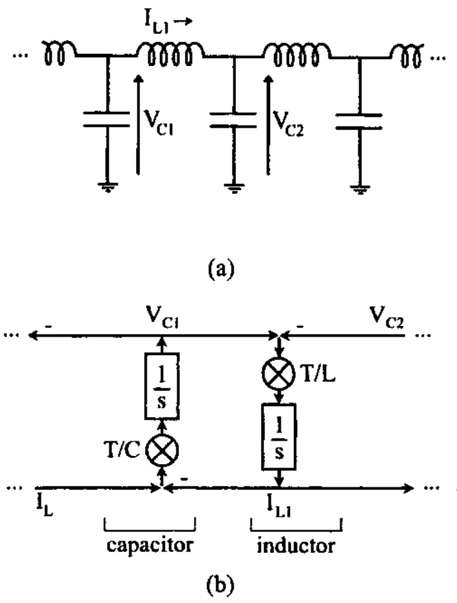

Figure 2: (a) One-Dimensional Inductor-Capacitor FDTD Equivalent (b) Signal Flow Graph. The '1/s' denotes Laplacian integration. Voltages are represented by the signals at the top of the signal flow graph, while currents are represented on the bottom.

Following Bruton's work [6], each integrator in Figure 2 is replaced by a 'lossless discrete integrator' (LDI) of the form:

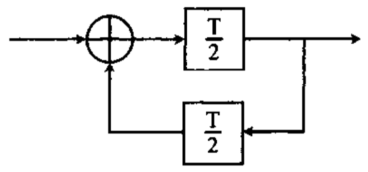

Figure 3: Lossless Discrete Integrator (LDI)

These integrators effectively 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 LDI's.

Finally, following Bruton's work [6] again, the delays are re-arranged to give the following signal flow graph:

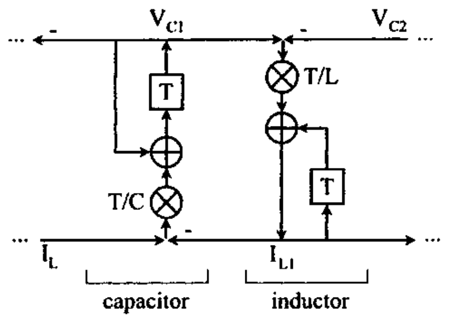

Figure 4: Signal Flow Graph of a One-Dimensional FDTD Cell

C. Hardware Implementation

Pipelined, bit-serial arithmetic was chosen in order to implement the signal flow graph of Figure 4. Bit-parallel arithmetic is the most common and familiar method for performing digital computations. In this case, all resultant bits of an operator are computed at once, in parallel. This method typically represents the fastest computation method and requires the most hardware. Bit-serial is at the opposite end of the spectrum. In this case, the result of an operator is computed one bit at a time. For a more detailed discussion of these concepts, the reader is referred to [7].

Pipelined bit-serial arithmetic was chosen for the following reasons. As discussed, the hardware cost of pipelined, bit-serial arithmetic units is low. Computational units are reused for each bit of system wordlength bits. Many computational units could be implemented in parallel for a fixed amount of hardware, because of their small size. The pipelined, serial nature allows for very short routing lengths, reducing hardware routing costs and simplifying the routing process.

Integer arithmetic was chosen in an attempt to 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. More research is necessary into the required (bit-wise) accuracy of coefficients, field values and the resulting stability and accuracy of the FDTD algorithm.

The atomic units of the bit-serial computational structure are: adders, subtractors, multipliers, shifters (multiply/divide by 2) and delays. A more complicated control structure is required to frame the system wordlength (SWL) groups of bits as they traverse the bit-serial structure. The signal flow graph in Figure 4 is converted into fixed-precision, integer computations on hardware using these pipelined-bit serial building blocks. The SWL was chosen to be 32-bits with 12-bit multiplier coefficients. Full details of the bit-serial hardware implementation are given in a paper the authors have written for the FPGA community [8].

D. Fixed Precision Implications

By experiment, using 8-bit coefficients to represent the T/C or T/L multiplier values, it was found that the resulting simulation was not bounded-input, bounded-output (BIBO) stable. 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.

E. One-Dimensional Resonator

A one-dimensional, free-space cavity resonator, ten FDTD cells in length and terminated in perfect electric conductors was implemented. A resonator represents a trivial example, but it is very useful for verification of the algorithmic implementation. Errors in the calculations quickly accumulate and the output becomes unbounded. 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 excitation is a short, time-domain impulse, intended to excite all frequencies within the resonator. The impulse is realized by biasing one of the capacitors with a non-zero value at the start of the simulation. Coefficients were chosen such that $\Delta x = 1.0$ cm and $\mu_r = \varepsilon_r = 1.0$ (converted to the inductor and capacitor values, respectively). Using 10 cells, this represents a resonator 10 cm in length.

III. Results

The target hardware device is the Xilinx Virtex Family FPGA, XCV300, and it offers 3,072 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 4-input lookup tables.

A. Simulation Results

The hardware FDTD computation, in one-dimension, successfully predicts fundamental and second harmonic resonant frequencies to within 0.65% and 1% of theoretical values. Similarly, a two-dimensional resonant structure predicts the first resonant frequency to within

1%. These predictions are identical to the results produced by an Intel-Linux computer running a traditional, C++, floating-point FDTD simulation.

B. Computation Speed and Hardware Requirements

The maximum operating frequency, for the one-dimensional case and reported by the Xilinx FPGA CAD tools, is 37.7 MHz. For a 32-bit SWL implemented in pipelined, bit-serial arithmetic, a new result is computed every 849 ns ($f_{operation}$ = 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 milliseconds. A one-dimensional FDTD computational cell occupies 86.5 Xilinx Virtex 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 two-dimensional FDTD cell requires 120 Virtex slices. Operating at a serial clock of 32 MHz and 40-bit SWL, new results are available every 1.25 microseconds (f_{op} = 0.8 MHz, 10,000 iterations = 12.5 milliseconds).

We can use this information to further predict the cost of three-dimensional computational cells. The one-dimensional case represents two fields. The two-dimensional cell representing three fields requires 120 slices, with the addition of two subtractors to combine additional fields into some calculations. Finally, it is predicted that the three-dimensional computational cell would require 265 slices to represent six fields.

IV. DISCUSSION

The one-dimensional FDTD algorithm has 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.

Simple calculations show that a typical three-dimensional simulation (100x100x100 = 1 million cells) would be too large to fit on a single hardware device. The largest part currently released by Xilinx offers 61,440 slices. Thus, it would take 4,313 of these devices to implement the entire simulation, at once, on hardware. From another perspective, the largest parts represent 10 million gate equivalents. With one million cells, there are 10 logic gate equivalents available per three-dimensional FDTD cell, per FPGA. This is clearly not enough to perform a useful *volume* computation. However, using this method, significant acceleration can be applied to one and two-dimensional analyses like transmission lines, waveguides and symmetrical simplifications of three-dimensional problems.

In order to achieve useful acceleration of the FDTD algorithm, other avenues are now being explored. A 10x10x10 FDTD cube could fit on five of the largest FPGA parts. We are now investigating methods for reusing a smaller computational engine to compute larger structures.

ACKNOWLEDGEMENTS

The authors would like to acknowledge partial support of this research by the National Science and Engineering Research Council (NSERC) of Canada.

REFERENCES

[1] Yee, K.S., "Numerical Solution of initial boundary value problems solving Maxwell's equations in isotropic media," *IEEE Trans. Antennas and Propagation*, Vol. 14, 1966, pp.302-307

[2] Taflove, Allen. Advances in Computational Electrodynamics – The Finite Difference Time Domain Method. Norwood, MA: Artech House Inc., 1998.

[3] Marek, J.R., Mehalic, M.A. and Terzuoli, A.J. "A Dedicated VLSI Architecture for Finite-Difference Time Domain Calculations" *8th Annual Review of Progress in Applied Computational Electromagnetics*, Monterey, CA, vol. 1, pp. 546-553, March, 1992.

[4] Grinin, S.V.. "Integer-Arithmetic FDTD Codes for Computer Simulation of Internal, Near and Far Electromagnetic Fields Scattered by Three-Dimensional Conductive Complicated Form Bodies", *Computer Physics Communications*, vol. 102, no. 1--3, pp.109-131, May, 1997.

[5] Gwarek, W.K.. "Analysis of Arbitrarily Shaped Two-Dimensional Microwave Circuits by Finite-Difference Time-Domain Method," *IEEE Trans. Microwave Theory and Techniques*, vol. 36, no. 4, pp.738-744.

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

[7] Hartley, R.I. and Parhi, K.K.. Digit-Serial Computation. Norwell, Massachusetts: Kluwer Academic Publishers, 1995.

[8] Schneider, R. N., Turner, L. E., and Okoniewski, M. M. "Application of FPGA Technology to Accelerate the Finite-Difference Time-Domain (FDTD) Method", *Tenth International Symposium on Field Programmable Gate Arrays*, February 24-26, 2002, Monterey, CA.