AN FPGA IMPLEMENTATION OF TWO-DIMENSIONAL FINITE-DIFFERENCE TIME-DOMAIN (FDTD) ALGORITHM

Wang Chen, Prof. Miriam Leeser, Prof. Carey Rappaport, Panos Kosmas

Department of Electrical and Computer Engineering, Northeastern University
wchen@ece.neu.edu, mel@ece.neu.edu, rappaport@ece.neu.edu, pkosmas@ece.neu.edu

Three-Dimensional Finite-Difference Time-Domain (3D FDTD) is a powerful method for modelling the electromagnetic field. The 3D FDTD buried object detection forward model is emerging as a useful application in mine detection and other subsurface sensing areas. However, the computation of this model is complex and time consuming. Implementing this algorithm in hardware will greatly increase its computational speed and widen its use in many other areas. We present an FPGA implementation to speedup the pseudo-2D FDTD algorithm which is a simplified version of the 3D FDTD model. The pseudo-2D model can be upgraded to 3D with limited modification of structure. We implement the pseudo-2D FDTD model and complete boundary conditions on an FPGA. The computational speed on the reconfigurable hardware is about three orders of magnitude faster than the software implementation.

Understanding and predicting electromagnetic behavior is more and more needed in key electrical engineering technologies such as cellular phones, mobile computing, lasers and photonic circuits [2]. After K. Yee first introduce the FDTD method in 1966, people began to realize its accuracy and flexibility for solving electromagnetic problems [1]. The FDTD method provides a direct time-domain solution of Maxwell's Equations in differential form by discretizing both the physical region and time interval using a uniform grid. Because this method can solve Maxwell's equations on any scale with almost all kinds of environments, it has become a powerful method for solving a wide variety of different electromagnetic problems [3].

However, the FDTD method was not used widely until the past decade when computing resources improved. Even today, The computational cost is still too high for real-time application of the FDTD method. To solve this problem, we present a reconfigurable hardware implementation of the 3D FDTD buried object detection forward model. This FDTD model was developed at Northeastern University for use in research on subsurface sensing of landmines via ground penetrating radar.

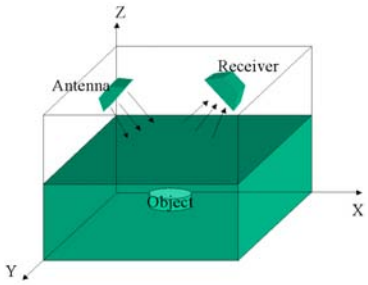


Fig. 1. 3D FDTD Buried Object Detection Forward Model Space

As shown in Figure 1, this model approximates a plane wave sent from ground penetrating radar with a 45° incidence angle, which is then fed into a three-dimensional space grid and propagated through an air-soil interface. As the wave is reflected from the boundary away from the location of the receivers, the possibility of detecting the small signal scattered from the buried object is high.

This model is computational intensive. The model space is discretized to up to millions of computational cells. For each of the cells, the FDTD algorithm updates all its parameters at every time step. Several hours may be needed to simulate 100 time steps to achieve useful information. What's more, the backward model, whose task is using the forward model's output data to detect the buried mines, runs the forward model iteratively to get the final result. So the running speed of the forward model is critical to the real-time application of the backward detecting device.

Implementation of FDTD in hardware will greatly increase its computational speed. With higher speed, the FDTD algorithm can be used in many other areas too. There are three methods we use to accelerate the algorithm:

1. Quantizing the 64-bit floating-point data to 30-bit fixed-point data while still achieving tolerable relative error.
2. Pipelining most of the calculations.
3. Parallelizing most of the pipelines to reduce processing time.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
20 AUG 2004	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
An FPGA Implementation of the Two-Dimensional Finite-Difference Time-Domain (FDTD) Algorithm	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Electrical and Computer Engineering, Northeastern University**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM001694, HPEC-6-Vol 1 ESC-TR-2003-081; High Performance Embedded Computing (HPEC) Workshop (7th)., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**25**	

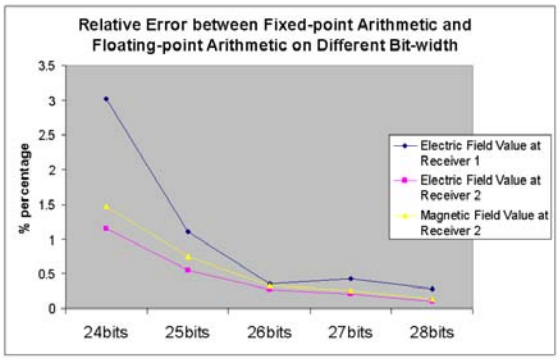


Fig. 2. Relative error between fixed-point arithmetic and floating-point arithmetic on different bit-width

The original algorithm uses a 64-bit floating-point representation which costs more hardware resources, consumes more power, and runs slower compare to using a fixed-point representation. Although the fixed-point representation has less dynamic-range, it fits the FDTD algorithm well since all the data in this algorithm are electromagnetic field values range between -1 and 1, and tend to be accurate to at most one part in 10,000. Figure 2 shows the relative error of different fixed-point bit-width data in the FDTD algorithm compared to floating-point. We chose the data structure with 26 bits after the binary point since this structure has an acceptable relative error and relatively short bit-width. In addition, one of the dimensions of the 3D model was set to 2 to create a pseudo-2D model. The pseudo-2D model is less complex and can be easily expand to the 3D model later, so we implemented this model first.

The hardware design accelerates the algorithm with pipelining and parallellism. All three electric and magnetic field-updating modules in the FDTD algorithm are pipelined and processed in parallel. The memory interface module, implemented on the FPGA chip using BlockRam, reads data from on-board memories and feeds them into the pipelines. All the processes are controlled by state machines. Since the FDTD algorithm has similar calculation and relatively regular structure, it is very suitable to be implemented using pipelining and parallelism.

Ideally, the more parallelism, the greater the speed. As long as there is sufficient FPGA chip area, we can implement more pipelines in parallel to speed up the design. In the FPGA chip we are currently using, a Xilinx Virtex-E, it is possible to use 6 or 12 pipelines instead of 3 pipelines to double or quadruple the processing speed.

The performance results of the software and hardware implementations are shown in Figure 3. The hardware design running on the FPGA chip is 24 times faster than fixed-point software running on a 3.0GHz PC and more than 100 times faster than the floating-point code.

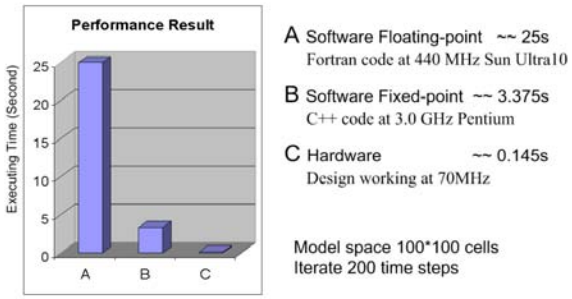


Fig. 3. Performance results - Softwares vs. FPGA Hardware

The FPGA hardware board we used is a Firebird Reconfigurable FPGA Computing Engine produced by Annapolis Micro Systems, Inc. It uses the Xilinx VIRTEX-E XCV2000E FPGA with over 2.5 million system gates.

1. REFERENCES

[1] Karl S. Kunz, Raymond J. Luebbers, "The Finite Difference Time Domain Method for Electromagnetics", *CRC Press, 1993*.

[2] Ryan N. Schneider, Laurence E. Turner, Michal M. Okoniewski, "Application of FPGA Technology to Accelerate the Finite-Difference Time-Domain (FDTD) Method", *FPGA 2002*.

[3] Kosmas, P., Wang, Y., and Rappaport, C., "Three-Dimensional FDTD Model for GPR Detection of Objects Buried in Realistic Dispersive Soil", *SPIE Aerosense Conference, Orlando, FL, April 2002, pp.330–338*.

An FPGA Implementation of the Two-Dimensional Finite-Difference Time-Domain (FDTD) Algorithm

Wang Chen
Panos Kosmas
Miriam Leeser
Carey Rappaport

Northeastern University
Boston, MA

Northeastern
UNIVERSITY

FDTD Algorithm and Implementation

- Finite Difference Time-Domain
  - Method for solving Maxwell's equations
  - Used for buried object detection
- Hardware Implementation
  - 3D to 2D model simplification
  - Data dependency analysis
  - Fixed-point quantization

Finite-Difference Time-Domain Method

- A direct time-domain solution of Maxwell's equations
- Accurate and flexible for solving electromagnetic problems
- Discretize time and electromagnetic space

Maxwell's Equations

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} - \sigma_m \vec{H} - \vec{M}$$

$$\nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} + \sigma_e \vec{E} + \vec{J}$$

$$\nabla \cdot \vec{D} = \rho_e$$

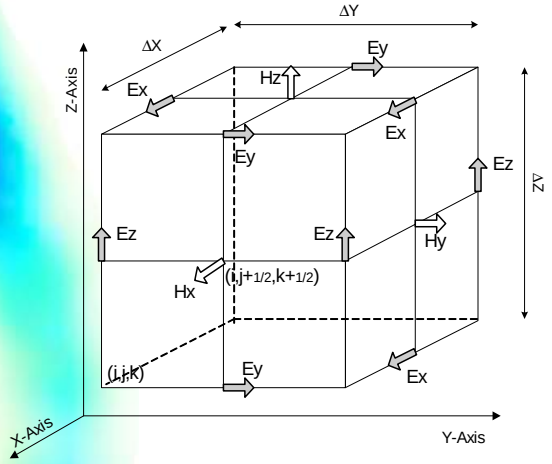
$$\nabla \cdot \vec{B} = \rho_m$$

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

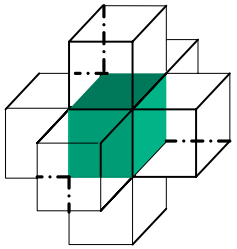
$$\vec{D} = \epsilon \vec{E}$$

FDTD Method (cont'd)

Yee Cell



Adjacent Cells



Taylor Series Expansion

$$\frac{\partial f(x_o)}{\partial x} = \frac{f(x_o + \Delta x) - f(x_o - \Delta x)}{2\Delta x} + O[(\Delta x)^2]$$

One FDTD Equation

$$\frac{\mu(j, j+1/2, k+1/2)}{\Delta t}[H_x^{n+1/2}(i, j+1/2, k+1/2)$$

$$-H_x^{n-1/2}(i, j+1/2, k+1/2)] = \frac{1}{\Delta z}[E_y^n(i, j+1/2, k+1)$$

$$-E_y^n(i, j+1/2, k)] - \frac{1}{\Delta y}[E_z^n(i, j+1, k+1/2)$$

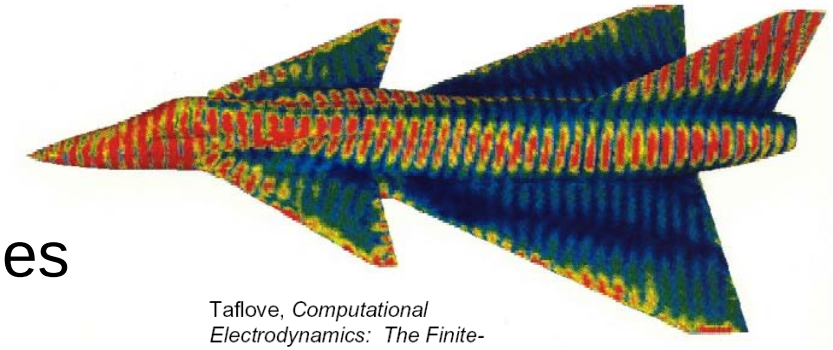
$$-E_z^n(i, j, k+1/2)] - \frac{\sigma_m(i, j+1/2, k+1/2)}{2}$$

$$\times[H_x^{n+1/2}(i, j+1/2, k+1/2) + H_x^{n-1/2}(i, j+1/2, k+1/2)]$$

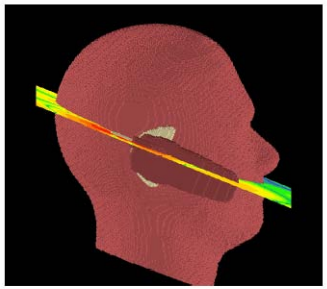
$$-M_x^n(i, j+1/2, k+1/2)$$

FDTD Applications

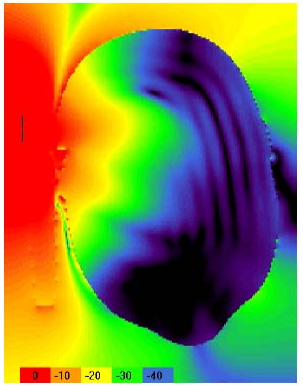
- Antenna Design
- Discrete Scattering Studies
- Medical Studies
  - The study of the cell phone electromagnetic waves' effect on human brain
  - The study of breast cancer detection using electromagnetic antenna



Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, 1995.

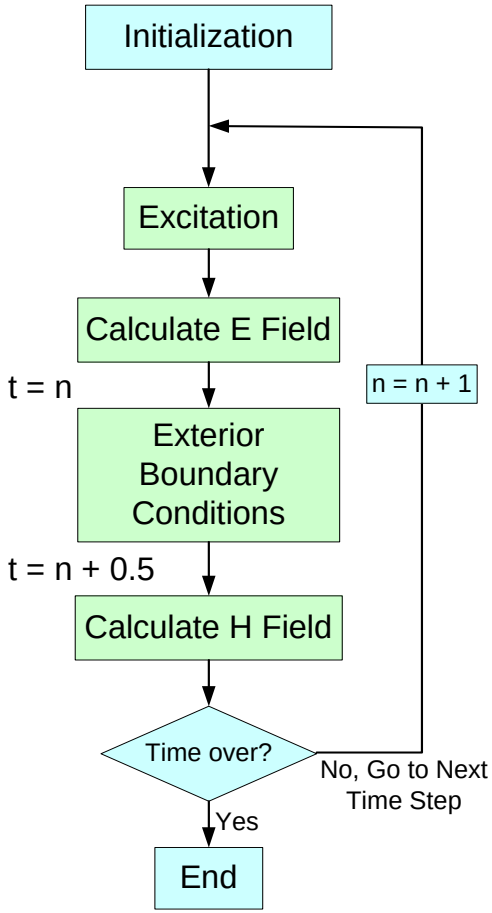


Cut plane through the cellphone

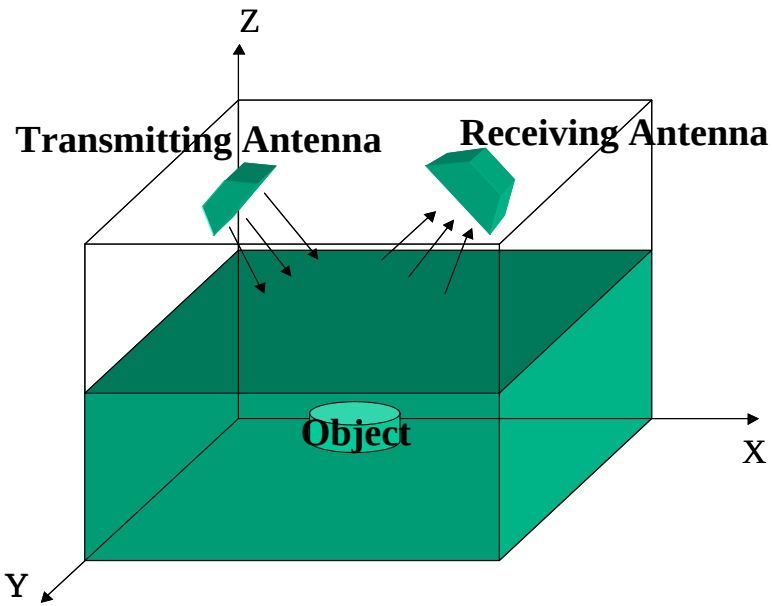


Remcom Inc. website: http://www.remcominc. com/html/index.html

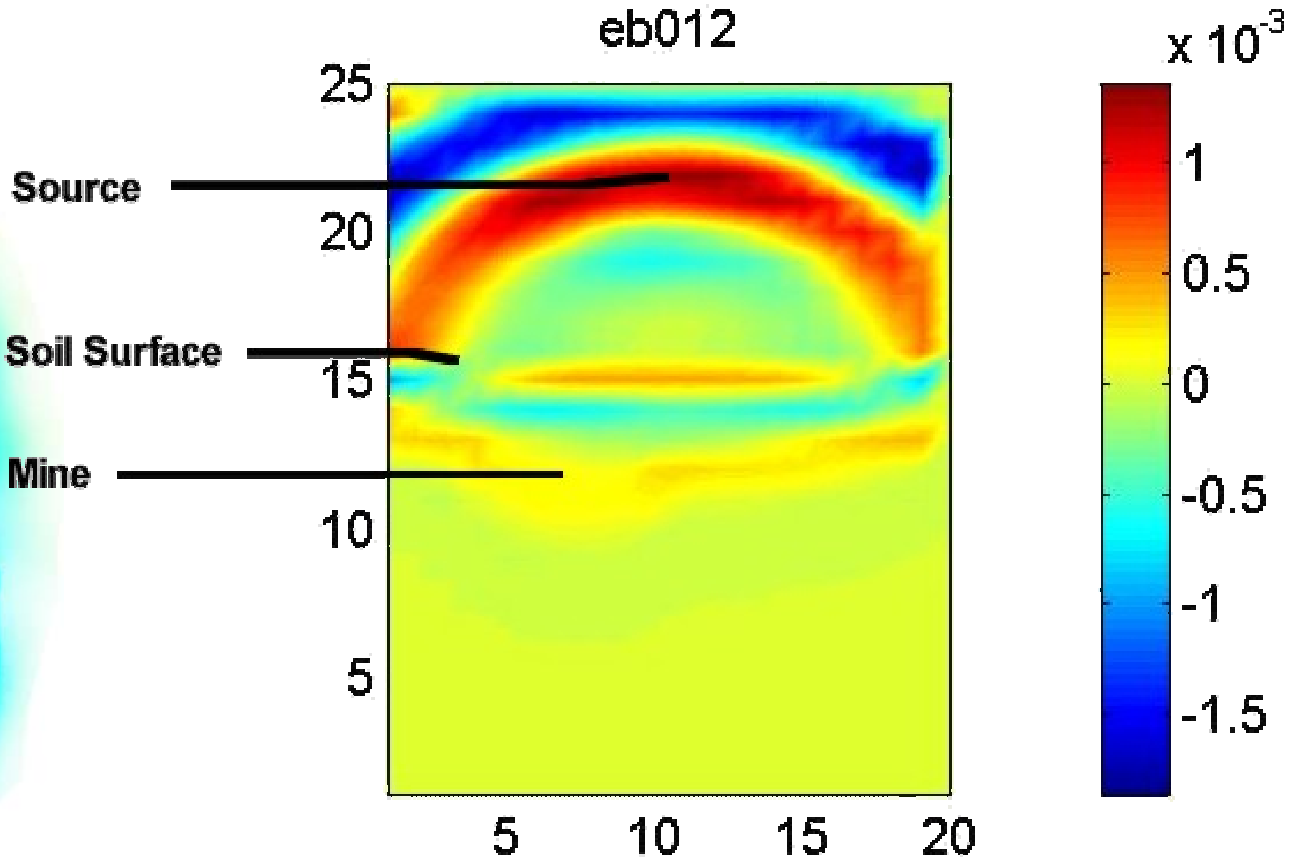
Buried Object Detection Forward Model



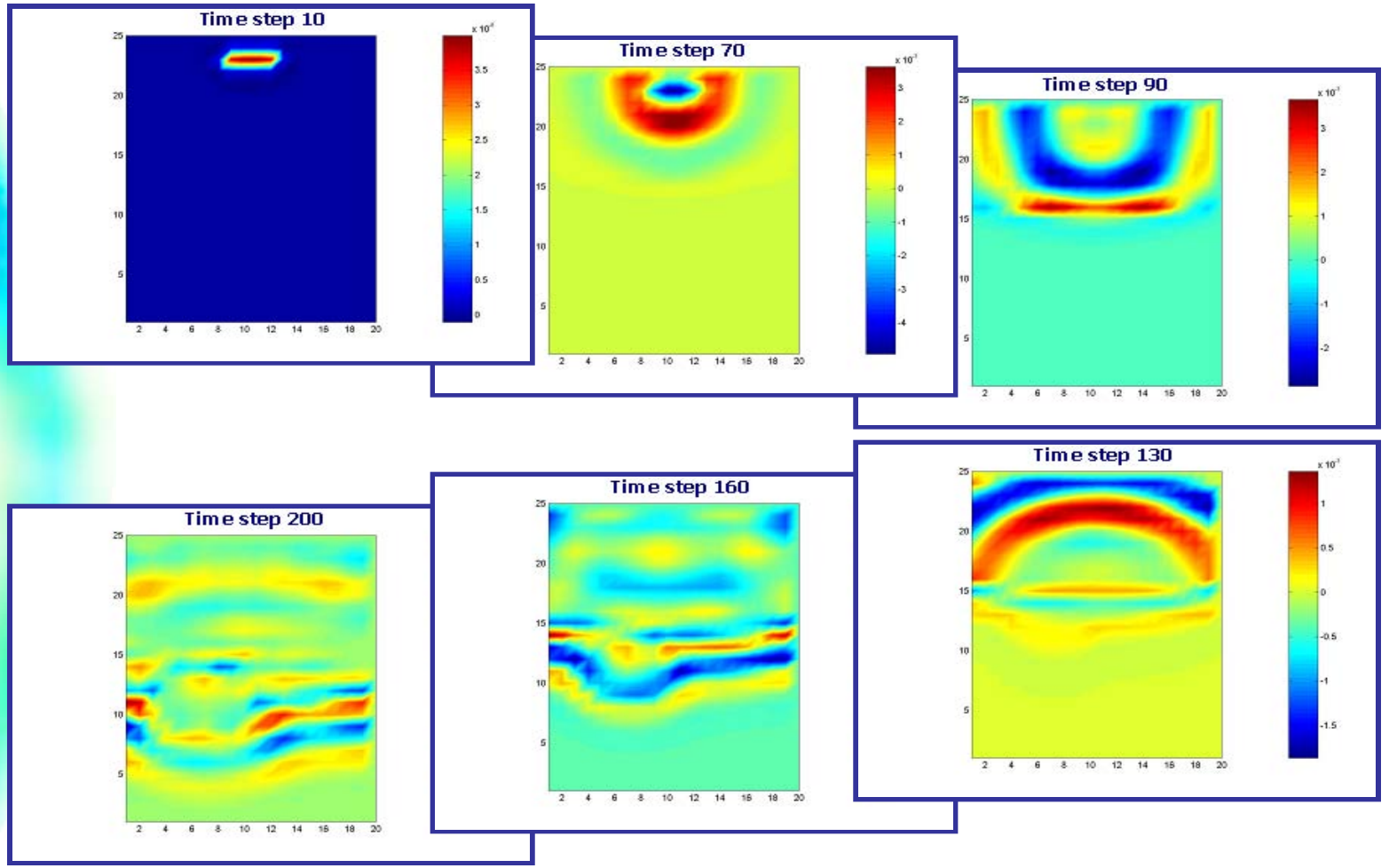
Buried Object Detection Model Space



FDTD Simulated Model Space



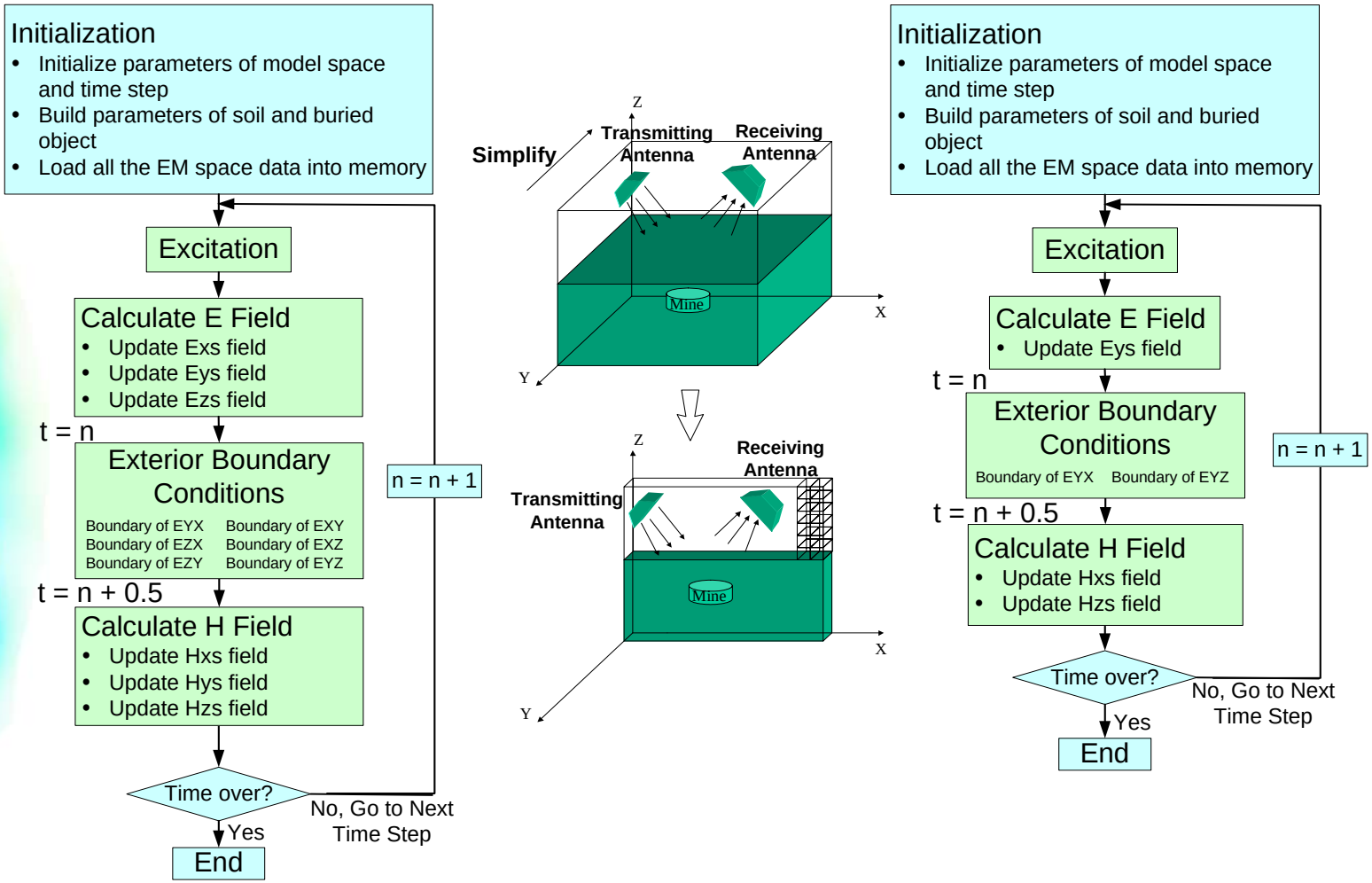
FDTD Simulated Model Space (cont'd)



Related Work

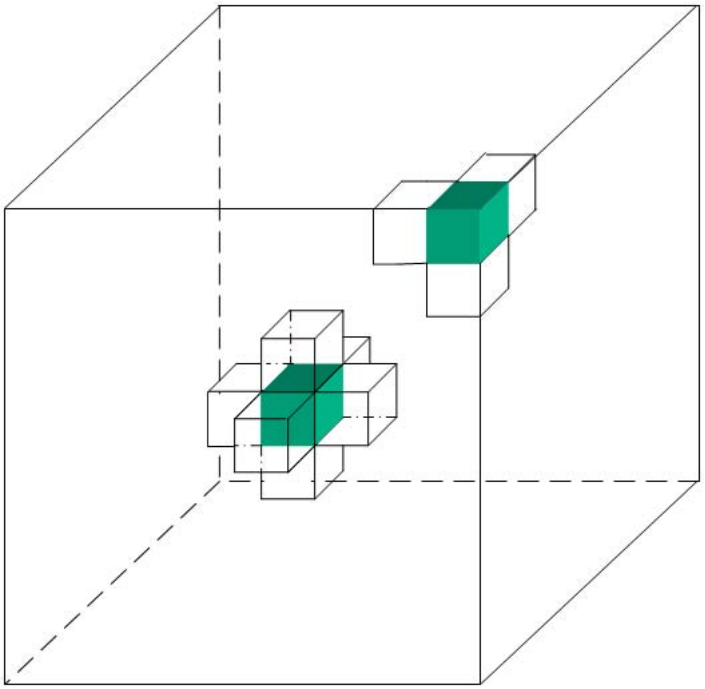
- Software acceleration of FDTD
  - Parallel computers do not provide significant speedup
- FPGA implementations of FDTD
  - 1D FDTD on hardware: architecture is too simple
  - Full 3D FDTD on hardware developed at UDel
    - Design is slower than software:
      - uses complex floating-point representation
      - no parallelism or pipelining
- Our 2D FDTD hardware implementation
  - 24 times speedup compare to 3.0G PC:
    - fixed-point representation
    - expandable structure

3D to 2D Model Simplification



Exterior Boundary Conditions

Mur-type Absorbing Boundary Condition



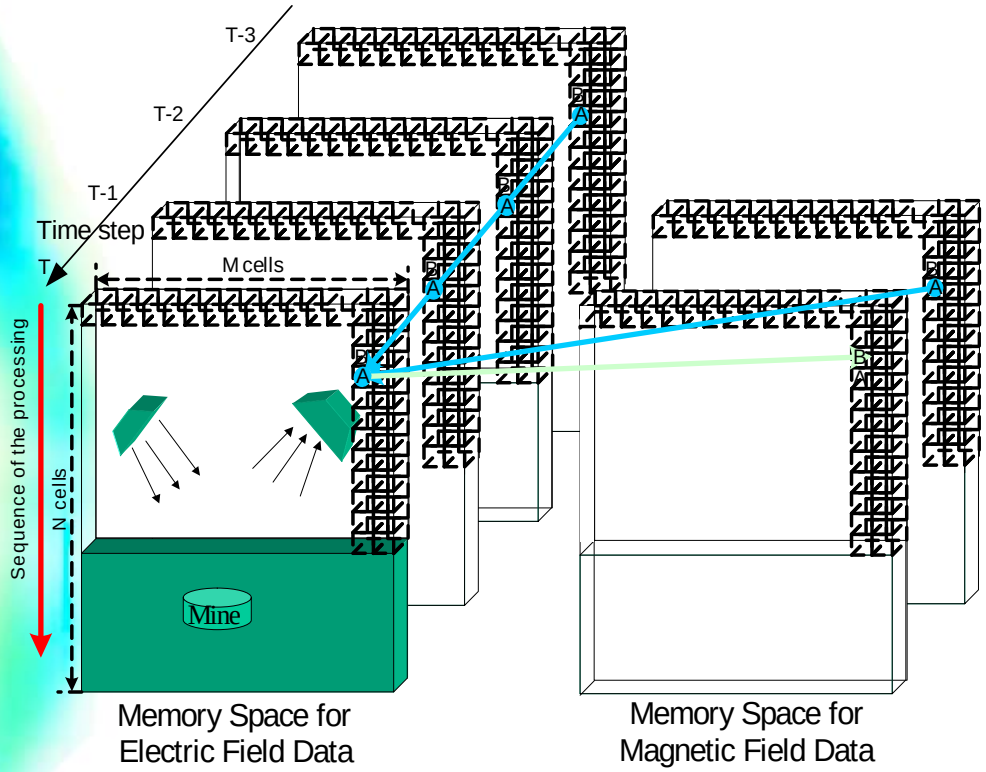
3D Model Space

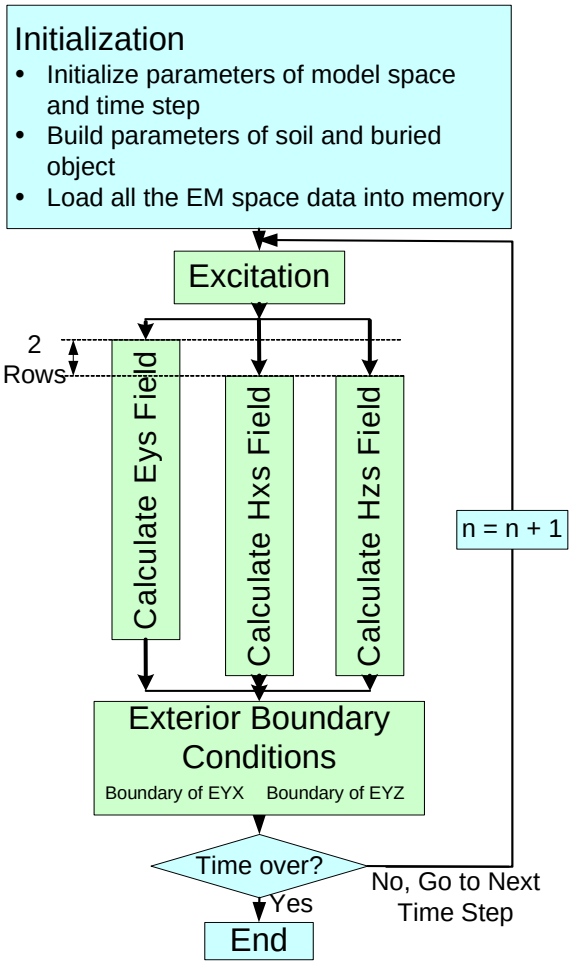
6 Faces and 12 Edges

2D Model Space

4 Edges

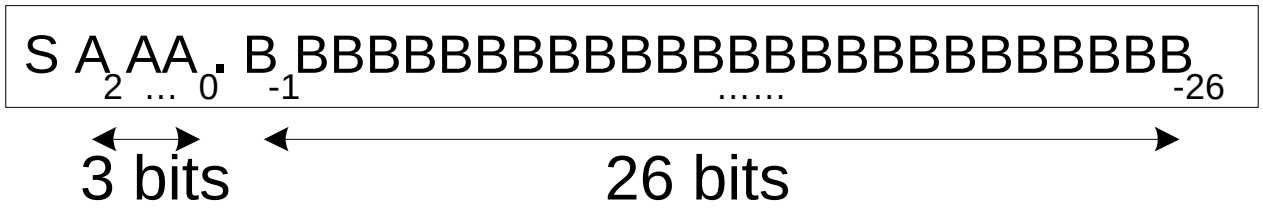
Data Dependency Analysis





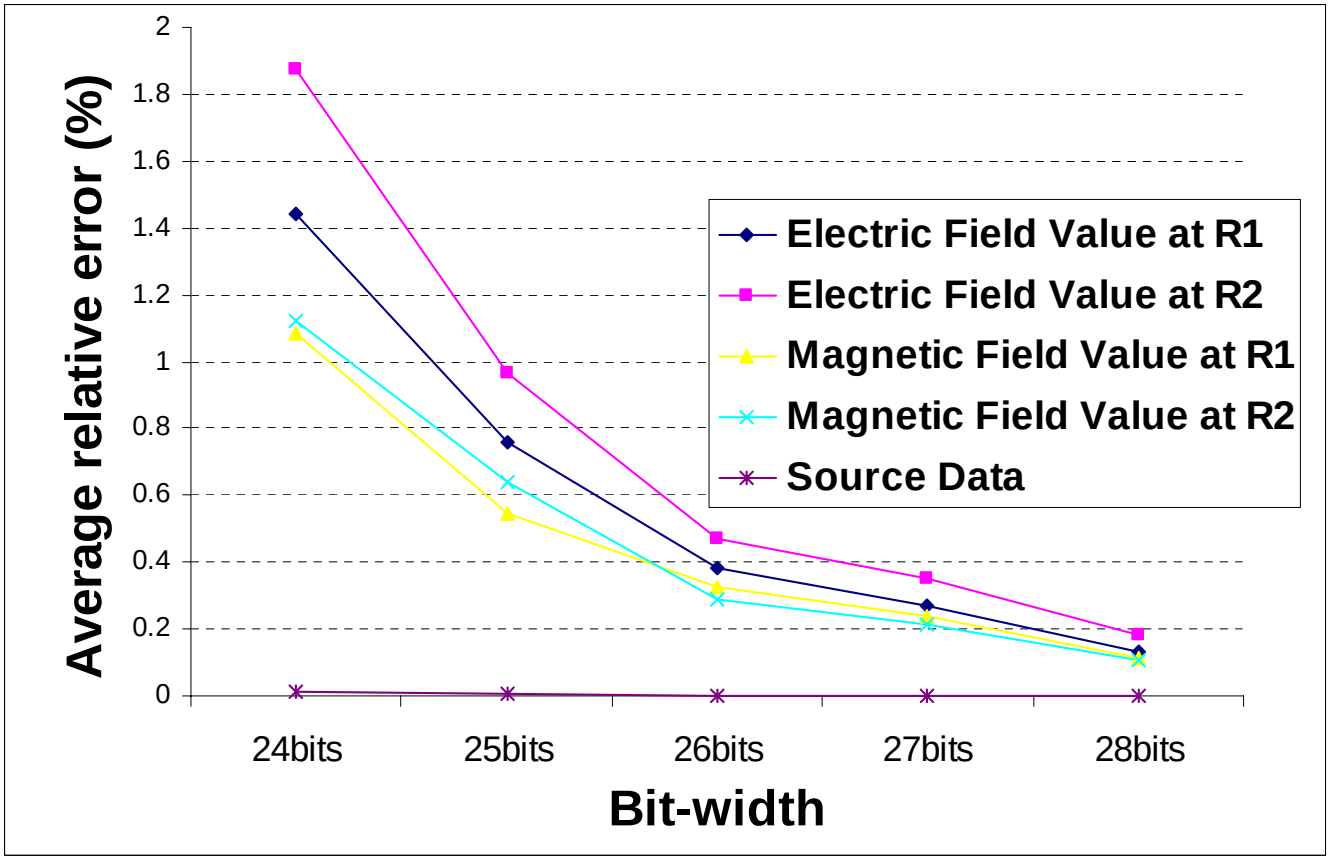
Hardware Acceleration

- Smart memory interface
- Parallelism
- Pipelining
- Quantized fixed point representation
  - Less area in datapath -- more parallelism
  - Careful error analysis to ensure accurate results

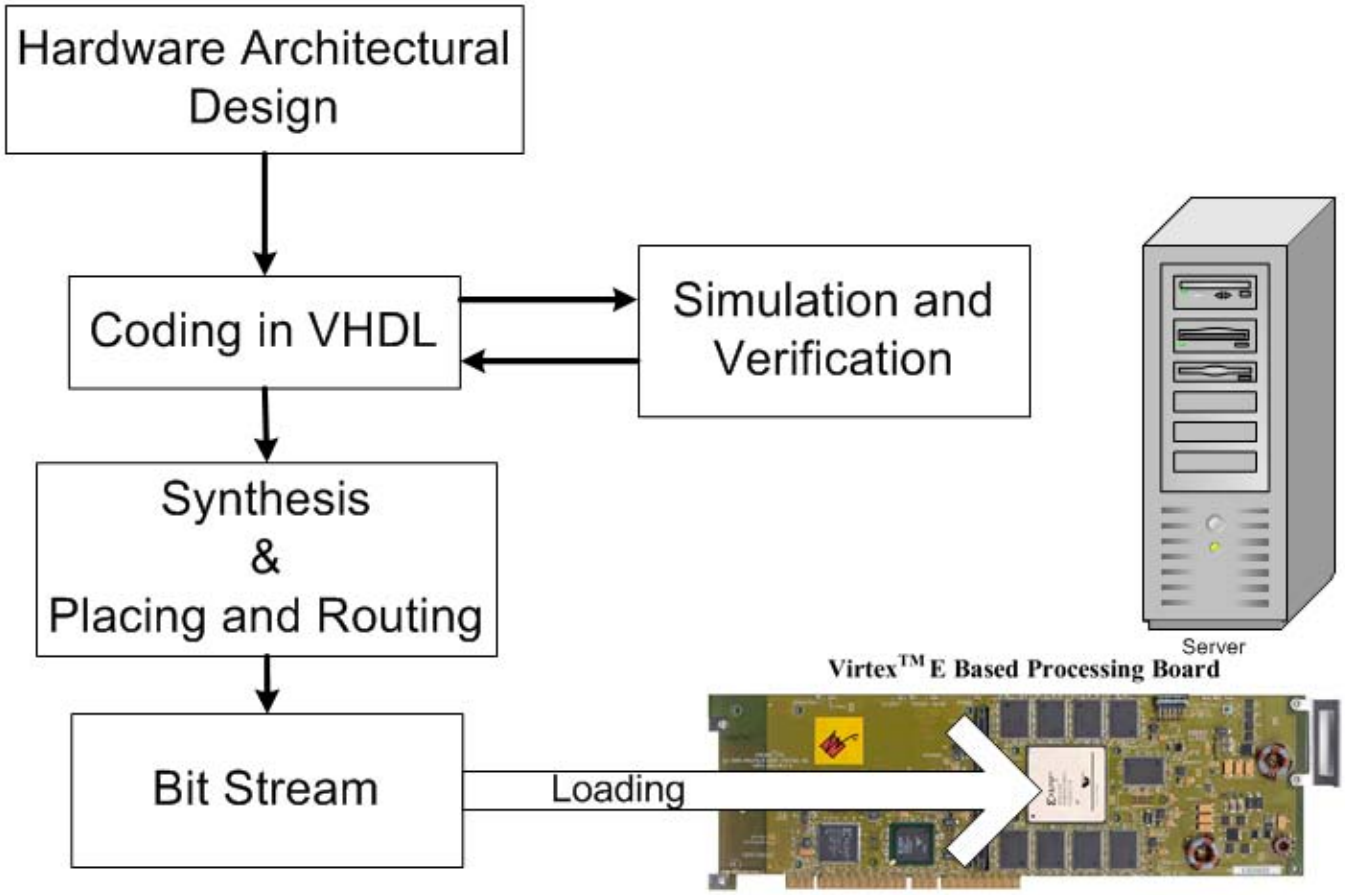


Fixed-point quantization

$$Relative\ error = \frac{|floating\ point\ data - fixed\ point\ data|}{|floating\ point\ data|}$$

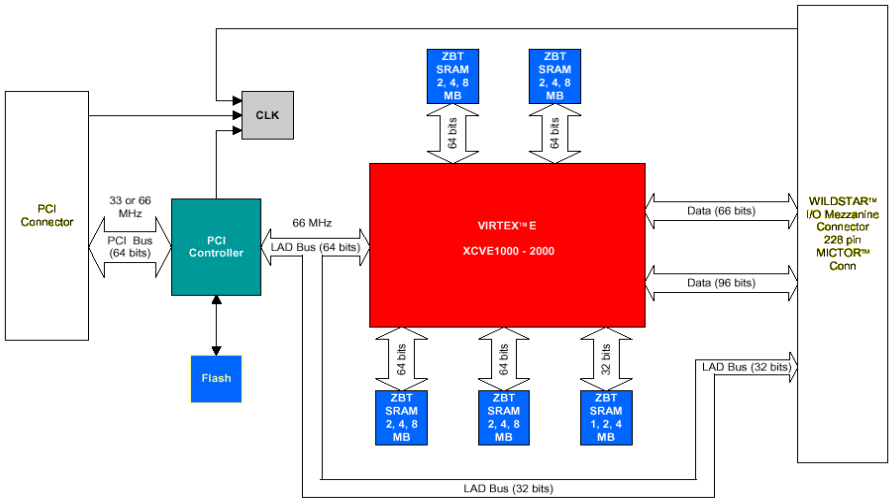


Design Flow


Hardware Architectural Design
Coding in VHDL
Simulation and Verification
Synthesis & Placing and Routing
Bit Stream
Loading
Server
Virtex™ E Based Processing Board

Firebird FPGA Board from Annapolis

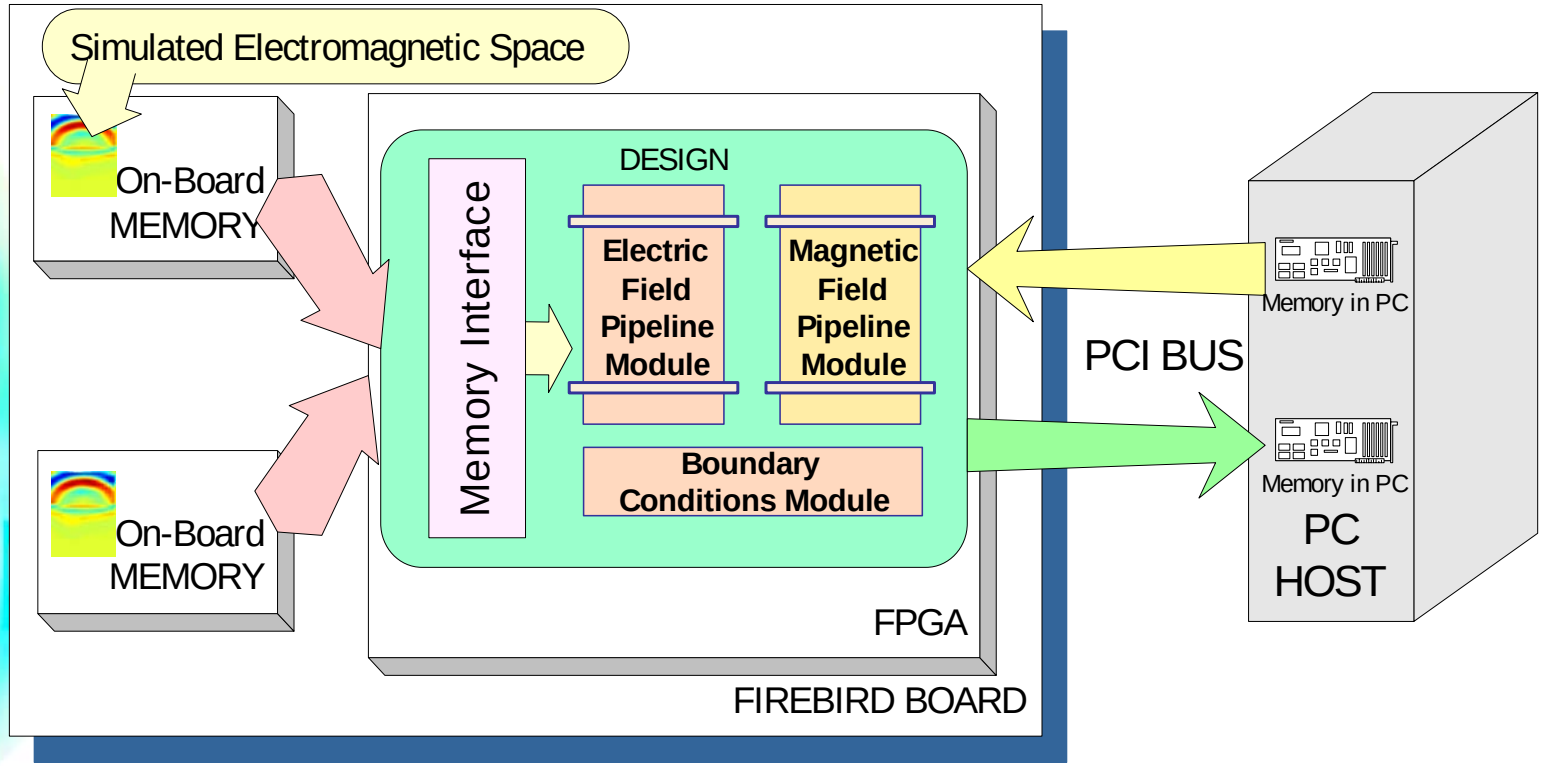
- A Xilinx VIRTEX-E XCV2000E with 2.5 million system gates
- Processing clock up to 150MHz FDTD runs at 70 MHz
- Five independent memory banks (4 x 64-bit, 1 x 32-bit) 288Mbytes in total
- 6.6Gbytes/sec of memory bandwidth
- 3Gbytes/sec of I/O bandwidth



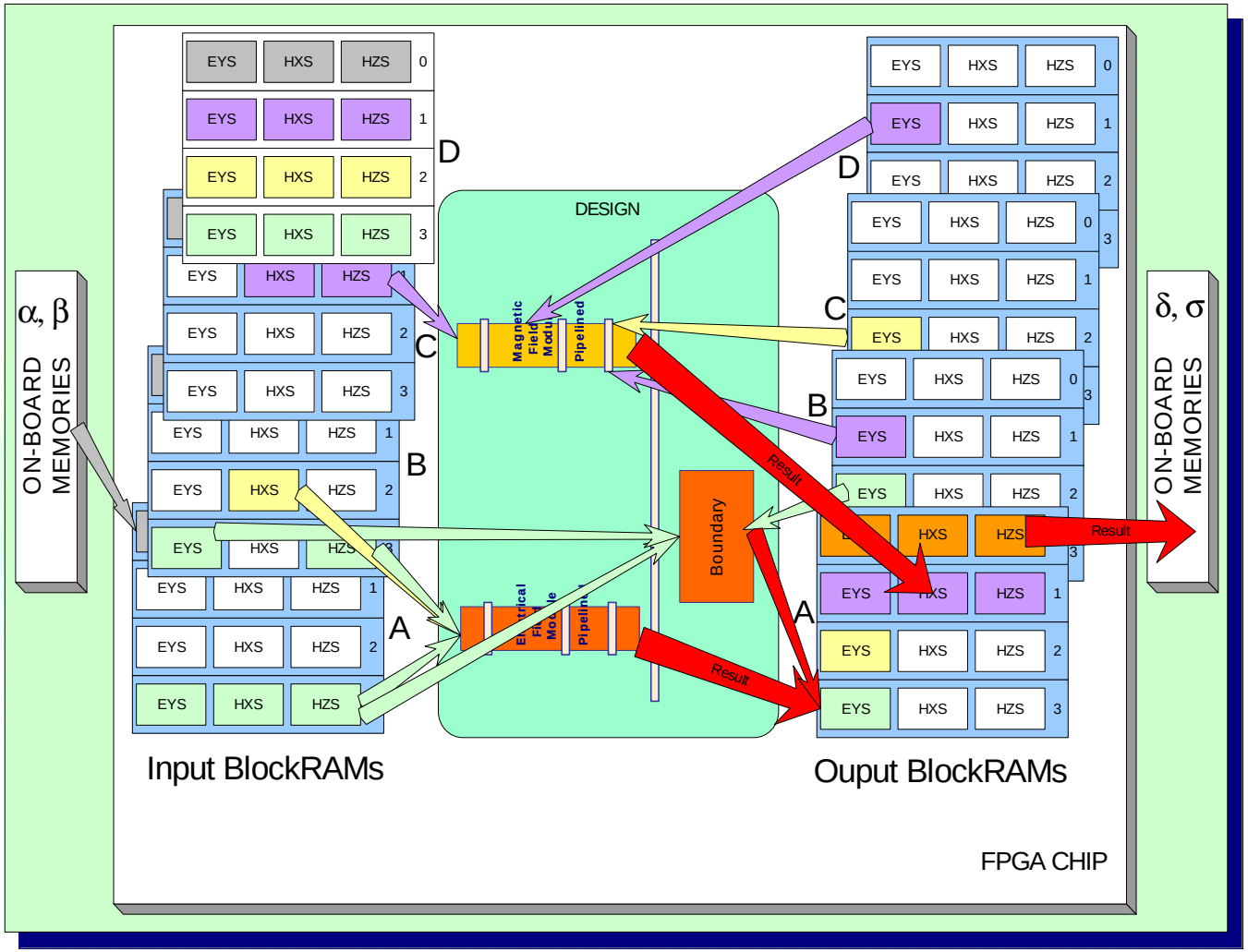
Utilization of Xilinx XCV2000E FPGA Chip

	Slices	BlockRAM
Number Available	19200	160
Number Used	8837	86
Percentage Used	46%	54%

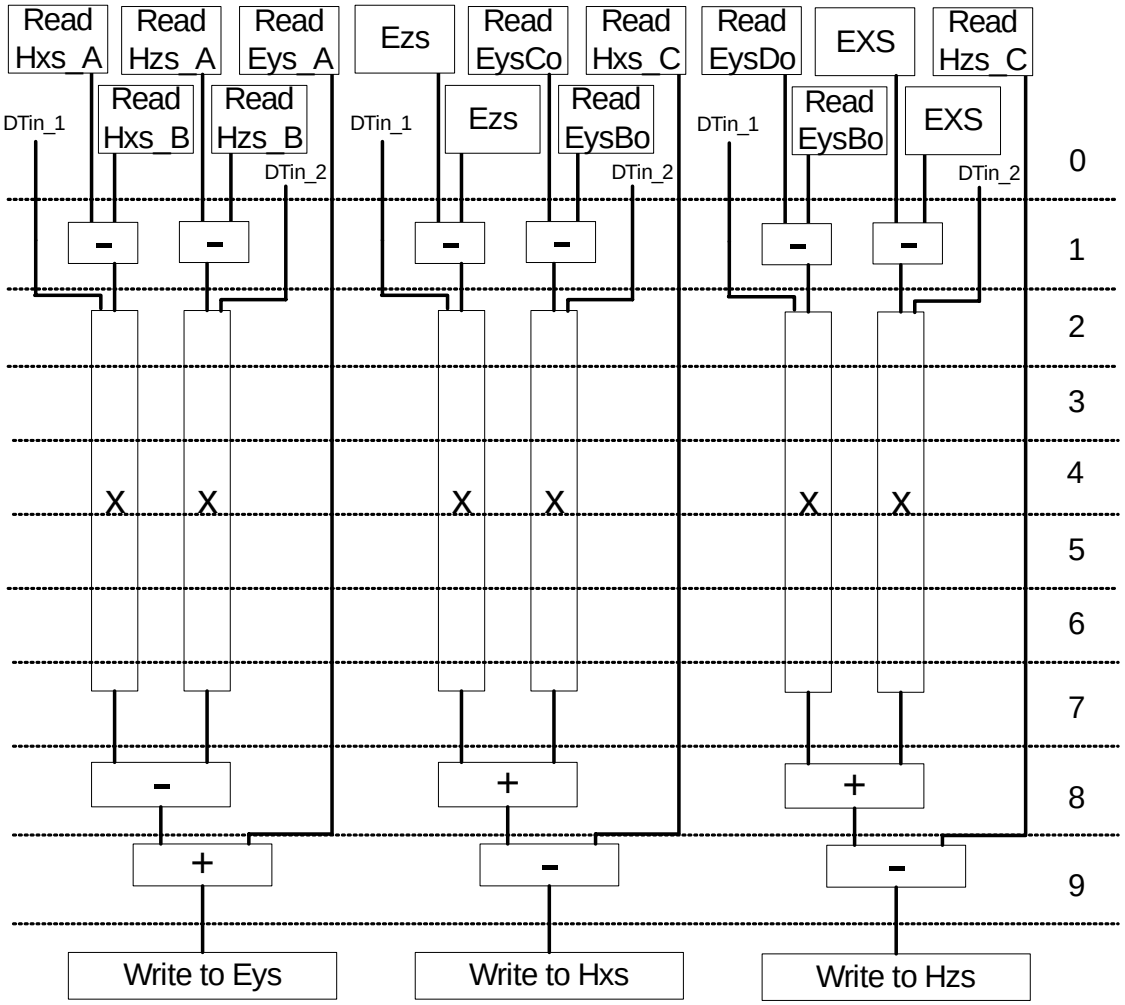
FDTD on Firebird Board



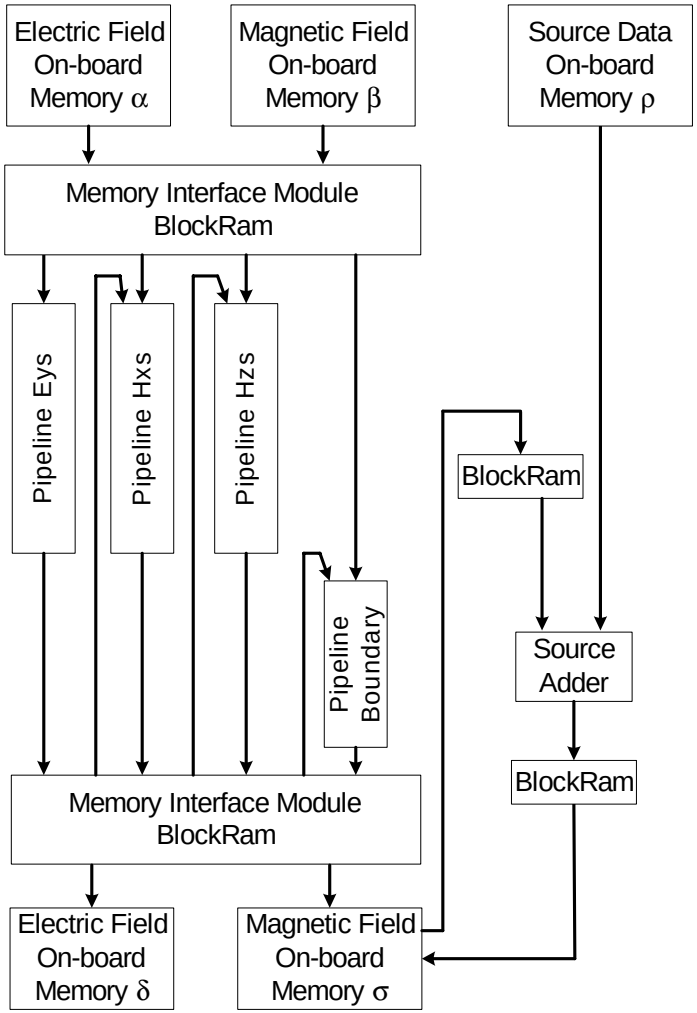
Memory Interface



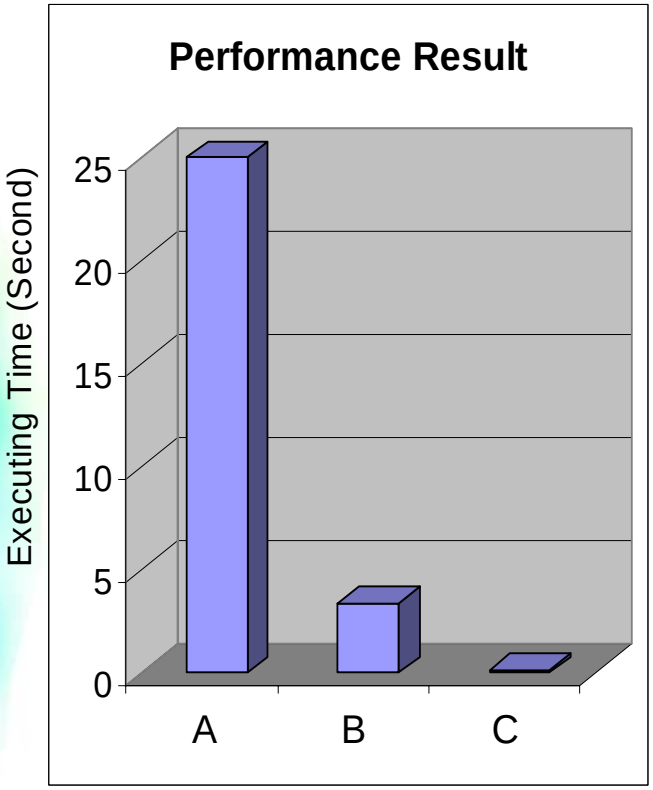
Pipelining and Parallelism



Data Flow


Electric Field On-board Memory α
Magnetic Field On-board Memory β
Source Data On-board Memory ρ
Memory Interface Module BlockRam
Pipeline Eys
Pipeline Hxs
Pipeline Hzs
Pipeline Boundary
BlockRam
Source Adder
BlockRam
Memory Interface Module BlockRam
Electric Field On-board Memory δ
Magnetic Field On-board Memory σ

Results and Performance



A Software Floating-point ~~ 25s
Fortran code at 440 MHz Sun Workstation

B Software Fixed-point ~~ 3.375s
C code at 3.0 GHz PC

C Hardware ~~ 0.145s
Design working at 70MHz

Model space 100*100 cells
Iterate 200 time steps

Conclusions

- FPGA Implementation of FDTD exhibits significant speedup compared to software: 24 times faster than 3GHz PC
- With larger FPGA, more parallelism will be available, hence more speedup
- Current design easily extendible to handle multiple types of materials, 3D space

Future Work

- Upgrade curent design to handle multiple types of materials
- Upgrade to 3D model space
  - Add three more field updating algorithms: same structure as the original three algorithms
  - Upgrade boundary condition updating algorithm
  - Redesign memory interface
- Apply FDTD Hardware to other applications