8

ELECTROMAGNETIC WAVE ANALYSIS USING FD-TD AND ITS IMPLEMENTATION ON THE CONNECTION MACHINE

A. T. Perlik and S. C. Moraites

8.1 Introduction

Finite-Difference Time-Domain (FD-TD) is a volumetric gridding time domain technique for solving Maxwell's electromagnetic field equations. Critical algorithm components were developed by Mur, Yee and others. The results of their efforts were brought to fruition by Taflove who published his first results on electromagnetic scattering in 1975, [1]. Through Taflove's efforts over the last decade, extending the basic algorithm to conformal surfaces, applying the technique to a variety of electromagnetic phenomena, and validating computational results against experimental data and other analytical approaches such as the method of moments, FD-TD has become a recognized EM computational technique by the electromagnetics community.

This chapter shows how FD-TD was mapped onto the Connection Machine (CM)†, a massively parallel Single Instruction Multiple Data (SIMD) computer architecture. A very brief overview of the FD-TD algorithm basics will be presented first. For an indepth treatment of the theory, the interested reader is referred to Chapter 8 in the second volume of this sequence [2]. Theoretical details are included here only for those algorithms that were newly developed. Next, an overview of the Connection Machine is in order. Following is a description of how FD-TD was mapped onto the CM. A section on algorithm testing and checkout is provided. The purpose is two-fold. Some of the testing provides insight into the FD-TD algorithm operation and its limitations. It is hoped that the discussion will help other investigators to understand how FD-TD works. Aside from the educational value on algorithm specifics, some parallel code debugging suggestions based upon our experience should prove useful. A set of illustrative problems is included which is complementary to those already published both in the literature and in the previous volume. It is appropriate to conclude with a discussion on developments that are underway.

a. Overview

Basically, FD-TD is used to model the propagation of an electromagnetic source in a finite volume of space containing an electromagnetic scatterer. See Fig. 8.1. A cubic cell spatial lattice grids the volume under study. Some researchers have employed computational grids

† "Connection Machine" is a registered trademark of Thinking Machines Corporation.

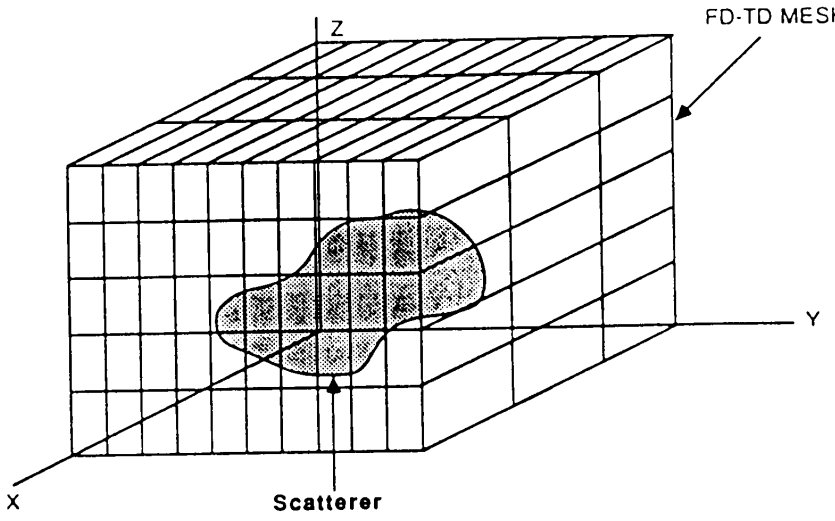

Figure 8.1 FD-TD computational geometry: scatterer embedded in a three-dimensional mesh.

that are stretched to match the surface contour of a scatterer. Unfortunately, runtime severely degrades and computational accuracy is not as great because the effect of numerical dispersion is more pronounced. Alternative gridding techniques still remain an area for continued research, however.

One particularly powerful feature of FD-TD is the ability to specify point properties of the scatterer under study at the subcell level. Aside from the obvious capability to study layered anisotropic scatterers, complex PEC structures can be analyzed too. For example, FD-TD was used to model a 72-element waveguide lens antenna structure (See Section 8.5b). Over 3.8 million unknown field components were determined over a regular cubic volumetric grid distributed inside each waveguide, along the waveguide walls and in the immediate near field. Although FD-TD employs volumetric gridding, it can be viewed as a thick layer technique for large classes of problems. Most practical scat-

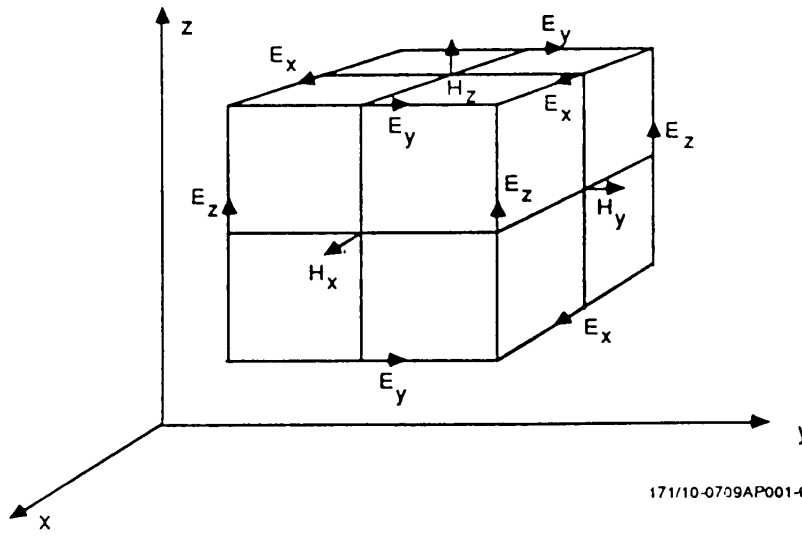

Figure 8.2 Yee lattice.

terers have a metallic skeleton if only for mechanical integrity. Provided that there are no reentrant features the inside need not be gridded or computed because the fields are zero. For problems that fall into this category gridding requirements grow with surface area not with volume. FD-TD has also been used [4] to provide correct solutions for three-dimensional eigenmodes in resonators.

An exploded view of a typical FD-TD cell appears in Fig. 8.2, the well-known Yee lattice. Note that E and H field components are computed on grids that are staggered by one-half cell. This formulation guarantees second order accuracy in the approximating difference equations. Another truly outstanding feature of this approach is that the discrete analog of the identities $\nabla \cdot \nabla \times (*) = 0$ and $\nabla \times \nabla(*) = 0$ are also valid [3], providing an excellent analytic check for physical correctness of the leap-frog or staggered grid formulation.

Two computational regions are established, a total field region and

a scattered field region. The traveling wave incident field gets added at the interface between them. The total field region completely encloses the scatterer. Application of boundary conditions at the scatterer surface is simplified because boundary conditions apply to total field quantities. Every effort is being made to preserve computational dynamic range, or accuracy. If a scattered field update were employed in the shadow region, the solution would be degraded with subtraction noise. A total field calculation is more accurate.

A problem solution is obtained by time stepping. The source of excitation can be a plane wave which travels through space and illuminates a scatterer. Alternatively, a local source can be defined and inserted into the computational grid. This technique is preferred to model a dipole exciting a horn antenna. If a time harmonic solution is required, time stepping continues until steady state is achieved for all field components in the grid. At steady state all field components are sinusoidal in time with spatially distinct converged magnitudes and converged relative phases. Far-field quantities are computed using a near-field to far-field Schelkunoff field equivalence theorem [5]. The number of time steps required to reach steady state depends upon the frequency of the incident wave as well as scatterer properties. Every field in the scatterer must "communicate" with all other fields in the scatterer. Two or more round trip sweeps through the scatterer are typically required. More round trips are needed near resonance. The proper test for convergence and concurrence of results between codes should be comparison of near fields or surface currents. Many results reported in the literature compare far-field data only. Unfortunately, there is no guarantee that far-field data agreement implies that the physics is accurately being modeled.

Update equations for the volumetric grid decompose into four categories: discretized Maxwell's equations for both E and H at interior grid points within the total field and scattered field regions, Mur update equations for E components at the interior points of grid termination planes, interpolation and extrapolation equations for calculating the fields at the edges of the computational rectangular grid, and equations for the incident fields. Boundary conditions are automatically satisfied for scatterers whose boundaries conform to the grid. Unfortunately, a staircase surface model is not sufficiently accurate for many design problems. A conformal surface code is currently under development that will predict near fields accurately even for scatterers

that have sharp points. Mur update equations are used to computationally terminate the volumetric grid. Basically, they are discrete versions of a one-way wave equation and are designed to "absorb" outgoing plane waves. In order to update a Mur boundary point to a new time, nearest neighbor spatial values both at the current time and at most two previous time steps are needed. Mur updates cannot be used to update fields along the terminating edges because there is insufficient nearest neighbor data. Interpolation and extrapolation techniques are used here. See [2] for details.

The version of FD-TD that is discussed in this chapter limits the scatterer geometry to either "staircase" or more generally "stairface" geometries. Staircase models assign entire FD-TD cells to either free space or to the scatterer. Maxwell's equations are point relationships between E and H vector fields and their material properties. Conductivity, for example is defined at a point. In every FD-TD cell a different conductivity can be associated with E_x, E_y, and E_z. More generally, each E field component represents the average E field value over a cell face. The material parameters ϵ, μ, σ, ρ can be viewed similarly. Rather than assigning entire FD-TD cells the same material properties, it is possible to assign material properties to each cell face, hence the descriptor "stairface model." Stairface models are extremely useful for modelling large PEC structures such as waveguide lens antennas, closed scatterers of any shape with or without coatings. Coatings in this version are limited to those that can be modeled with cell face resolution. Most of the limitations found in the FD-TD code used here were limitations of the particular code version and not of FD-TD in general. Figure 8.3 illustrates the difference in field assignment between staircase and stairface modeling for a section of a two-dimensional scatterer.

Taflove has demonstrated an accurate conformal surface extension to two-dimensional scatterers, singly curved three-dimensional ones and most recently spheres. Work is currently underway both at Northwestern University and at MRJ to develop a full doubly-curved three-dimensional conformal surface model for PEC scatterers that have sharp points such as the NASA almond [6]. The basic rectangular grid remains intact. Cell faces cutting through the scatterer surface are handled in a special way. Results of this work are currently being validated and will be reported at a later date.

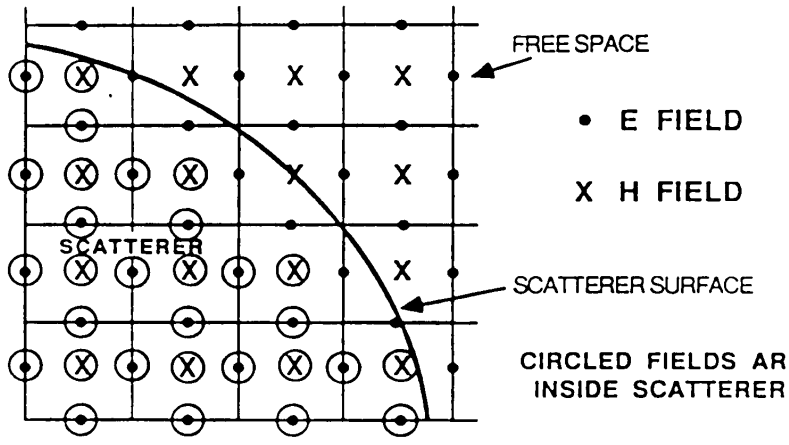

Figure 8.3a Two-dimensional staircase model.

b. Boundary Conditions at Interfaces

One of the reasons why Maxwell's equations are difficult to solve is that they are a set of partial differential equations with discontinuous coefficients. The nature of this discontinuity needs to be understood and modeled correctly. For materials in which the spatial variation of fields is gradual enough to be properly and efficiently sampled using a uniform gridding no special techniques beyond standard FD-TD are needed. However, when field variations change rapidly over the span of a cell, then building special field surface models for more precisely computing surface fields—even for rectangular parallel-piped geometries—is useful. The development below illustrates how a more accurate field surface model can be derived.

Consider two adjacent cell faces as illustrated in Fig. 8.4. Assume that the scatterer boundary lies along the common edge. For simplicity a two-dimensional discussion is conducted. Results apply for three

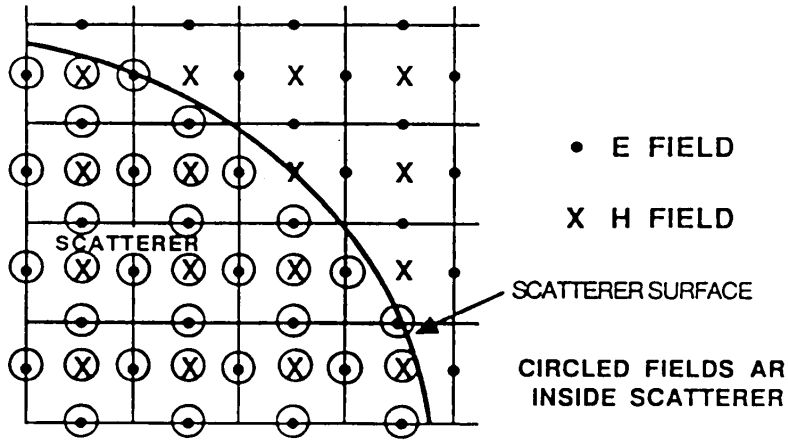

Figure 8.3b Two-dimensional stairface model.

dimensions as well but the added geometric complexity would provide no additional physical modeling insight. Using the quasi-static interpretation of the field component update process which has been shown to be valid on a time step per time step basis, [2], Faraday's Law can be applied to each cell face for determining the labelled H fields. Both the left and right cell face H fields can be updated using the material properties assigned to each face and the E fields assigned to each edge. Since E_{TAN} is continuous, the same tangential E field is used in both H updates even though there is a material discontinuity across the cell faces. Figure 8.5 illustrates the quasi-static model that is needed to update E_{TAN}.

Following the quasi-static interpretation in [2], an out of plane Ampere's Law can be applied to the cell face with E_{TAN} at the center and assumed constant throughout the cell face. The Yee grid provides this cell face leap-frog interpretation, but note in this case that the

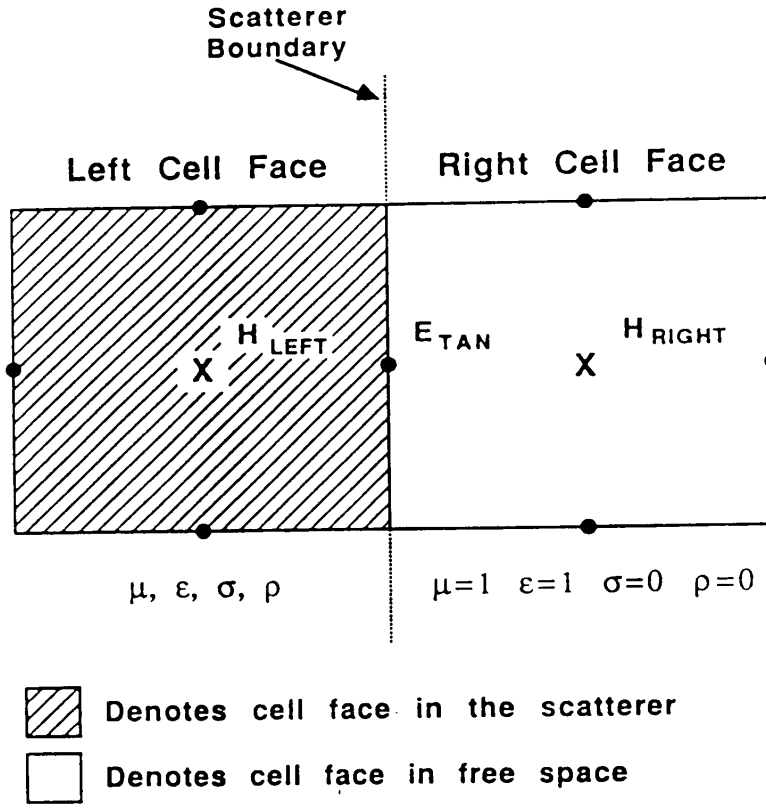

Figure 8.4 *H* **Field updates at scatterer surface. In plane Faraday's law can be applied to both left and right faces.**

scatterer boundary partitions the cell face. Consequently, the EM parameters are no longer homogeneous over the entire region to which Ampere's Law is applied.

One approach for modeling the parameter discontinuity is to simply incorporate the parameter discontinuity into the integral:

$$\int_{\text{cell face area}} \epsilon E_{TAN} \cdot dA = \int_{\text{left face area}} \epsilon_{left} E_{TAN} \cdot dA + \int_{\text{right face area}} \epsilon_{right} E_{TAN} \cdot dA \tag{1}$$

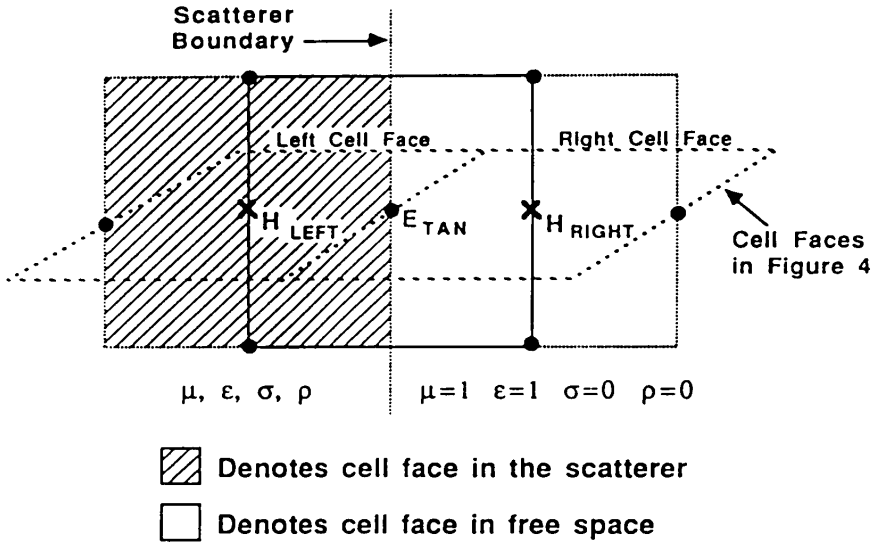

Figure 8.5 E_{TAN} **field update at scatterer surface. Out of plane Ampere's law is applied, but discontinuities in EM parameters must be included in the analysis.**

but an even more accurate model can be introduced with only a little additional effort. Consider updating the surface E field at a lossy half-space. Figure 8.6 provides the pictoral support where the spatial variation of the E field is sketched near the surface of a lossy half-space. A linear field variation is assumed for a distance $\lambda/40$ to $\lambda/20$ just outside the scatterer. Inside the decay is assumed exponential:

$$E_y(x) = E_{yTAN}\, e^{-|x|/\delta} \tag{2}$$

with E_{yTAN} the surface field signed magnitude, and skin depth $\delta = 1/\sqrt{\pi f \mu \sigma}$ for large σ, i.e., $\delta \ll \Delta x$.

Strictly speaking, the field variation inside the medium is a damped

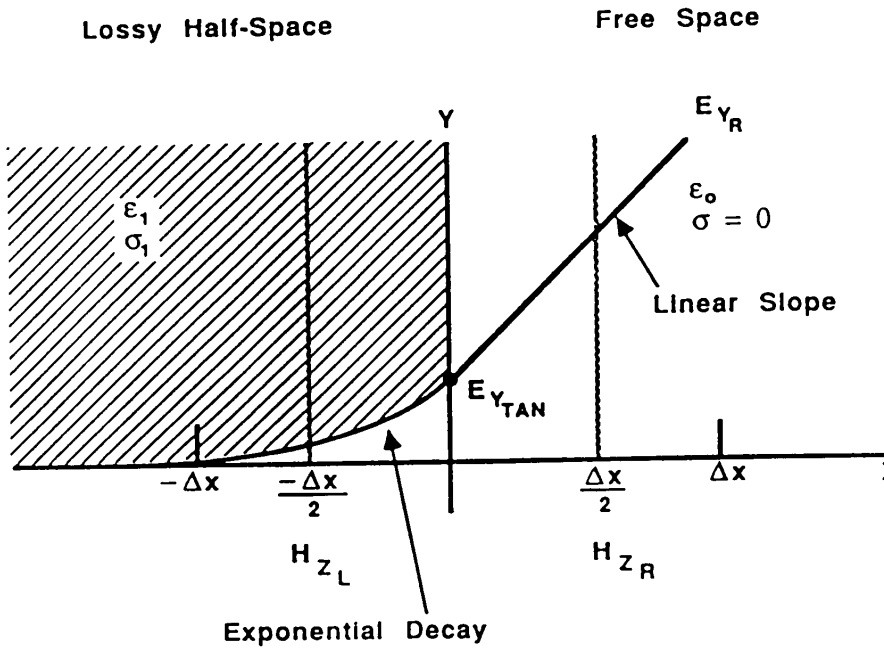

Figure 8.6 Tangential E Field variation for a lossy half-space ($\lambda/20$ gridding).

sinusoid (see Fig. 8.7), but the assumed exponential decay (envelope only) should not produce a significant error since it is only explicitly used in the formulation for materials too lossy to be correctly modeled with standard cells.

Ampere's Law will now be applied using the field variation provided in Fig. 8.6:

$$\oint H \cdot dI = \int J \cdot dA + \frac{\partial}{\partial t} \int D \cdot dA$$

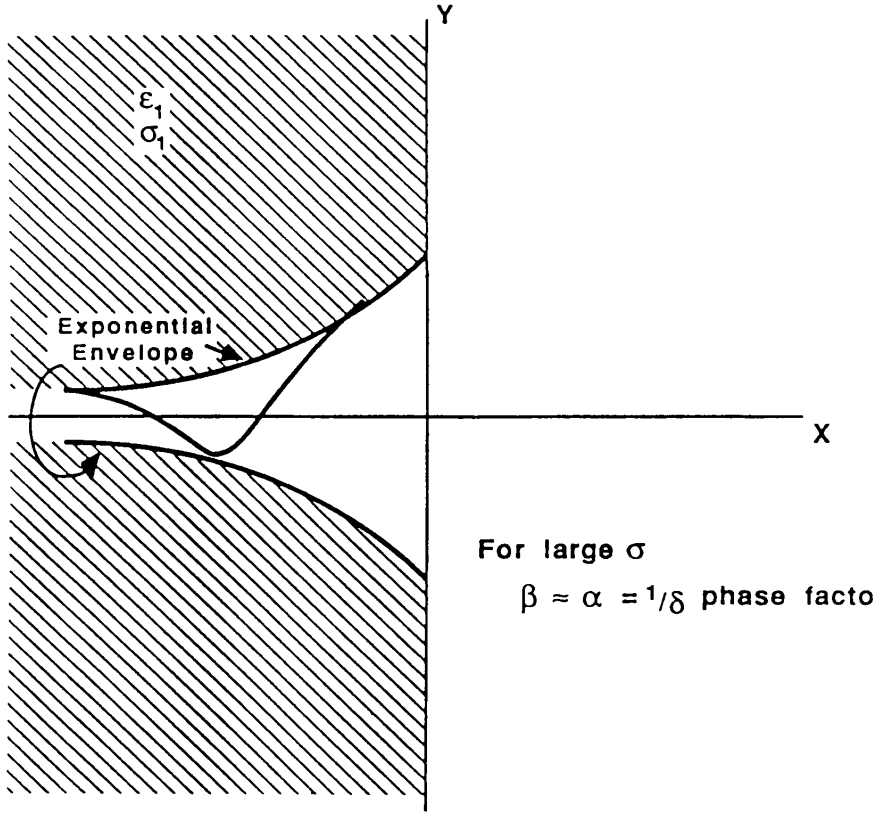

Figure 8.7 Field decay in a lossy medium.

$$(H_{z_L}^n - H_{z_R}^n)\Delta z = \left[\int_{-\Delta x/2}^{0} \sigma_L E_{y_{TAN}}^n e^{x/\delta} dx\right]\Delta z + \frac{\partial}{\partial t}\left[\int_{-\Delta x/2}^{0} \epsilon_L E_{y_{TAN}}^n e^{x/\delta} dx\right]\Delta z + \frac{\partial}{\partial t}\left\{\int_{0}^{\Delta x/2} \epsilon_0 \left[E_{y_{TAN}}^n + \left(\frac{E_{y_R}^n - E_{y_{TAN}}^n}{\Delta x}\right)x\right] dx\right\}\Delta z \tag{3}$$

where n is the time step index. Integrating and simplifying:

$$(H^n_{z_L} - H^n_{z_R}) = \left[\sigma_L\left(\frac{E^{n+1/2}_{yTAN} + E^{n-1/2}_{yTAN}}{2}\right) + \epsilon_L\left(\frac{E^{n+1/2}_{yTAN} + E^{n-1/2}_{yTAN}}{\Delta t}\right)\right]\left[\delta e^{x/\delta}\right]^{x=0}_{x=-\frac{\Delta x}{2}} + \frac{\partial}{\partial t}\left(\epsilon_0(3E^n_{yTAN} + E^n_{yR})\frac{\Delta x}{8}\right) \tag{4}$$

Approximating the partial time derivatives with:

$$\frac{\partial}{\partial t}\left(E^n_{yTAN}\right) \approx \frac{E^{n+1/2}_{yR} - E^{n-1/2}_{yTAN}}{\Delta t} \tag{5a}$$

and

$$\frac{\partial}{\partial t}\left(E^n_{yR}\right) \approx \frac{E^{n+1/2}_{yR} - E^{n-1/2}_{yR}}{\Delta t} \tag{5b}$$

and solving for $E^{n+1/2}_{yTAN}$ yields:

$$E^{n+1/2}_{yTAN} = \frac{\frac{3}{8} - \frac{\delta}{\epsilon_0\Delta x}\left(\frac{\sigma_L\Delta t}{2} - \epsilon_l\right)(1 - e^{-\Delta x/2\delta})}{\frac{3}{8} + \frac{\delta}{\epsilon_0\Delta x}\left(\frac{\sigma_L\Delta t}{2} + \epsilon_L\right)(1 - e^{-\Delta x/2\delta})} E^{n-1/2}_{yTAN} - \frac{1}{8\left[\frac{3}{8} + \frac{\delta}{\epsilon_0\Delta x}\left(\frac{\sigma_L\Delta t}{2} + \epsilon_L\right)(1 - e^{-\Delta x/2\delta})\right]}(E^{n+1/2}_{yR} - E^{n-1/2}_{yR}) + \frac{\Delta t}{\epsilon_0\Delta x\left[\frac{3}{8} + \frac{\sigma}{\epsilon_0\Delta x}\left(\frac{\sigma_L\Delta t}{2} + \epsilon_L\right)(1 - e^{-\Delta x/2\delta})\right]}(H^n_{z_L} - H^n_{z_R}) \tag{6}$$

As σ goes to infinity, δ goes to zero like $1/\sqrt{\pi f\mu\sigma_L}$. The coefficient of

$$E^{n-1/2}_{yTAN}$$

approaches -1 from above. It is also easy to see that the coefficients of:

$$(E^{n+1/2}_{yR} - E^{n-1/2}_{yR})$$

and

$$(H^n_{z_L} - H^n_{z_R})$$

go to zero as well. Equation (6) is therefore stable. Even if the initial value of

$$E_{yTAN}^{-1/2}$$

is nonzero, the recursion relation will converge to zero for a PEC as it should.

c. Magnitude and Phase

In its present formulation, FD-TD time steps instantaneous values for all three field components of both E and H. The scattering problems of interest are ones where single-frequency sinusoidal plane wave excitation bathes a scatterer whose makeup consists of linear time-invariant materials. Consequently, at steady state all E and H field components are sinusoidal in time at the incident plane wave frequency. In order to compute far-field quantities the tangential fields over a closed surface containing the scatterer are multiplied by the free space Green's function and integrated. A Schelkunoff field equivalence theorem can be invoked to justify the calculation. Integration over this Schelkunoff surface requires knowledge of both the magnitude and phase of all tangential fields. A similar computation is required to compute surface currents and to identify when steady state is reached.

In general, a typical field component at steady state has the representation:

$$g_{A,f,\phi}(t) = A\sin(2\pi ft + \phi) + B \tag{7}$$

where

t	is time
f	is the incident plane wave frequency
A	is the unknown field component amplitude
ϕ	is the unknown field component phase
B	is the bias value which is theoretically equal to zero.

Using two instantaneous field values, a simultaneous system can be derived to solve for both $\sin(\phi)$ and $\cos(\phi)$. This produces a value for ϕ with the proper quadrant information. Once ϕ is determined, A can be obtained. Unfortunately, the technique is susceptible to high frequency noise and any numerically generated slowly varying bias. An alternative technique was developed to mitigate these problems.

The amplitude of each field component is computed using a discrete expression for root mean square (RMS) and multiplying by $\sqrt{2}$ to obtain the amplitude of the sinusoid:

$$A_g = \sqrt{2}\left\{\frac{1}{N}\sum_{n=1}^{N}\left(g(n) - \frac{1}{N}\sum_{m=1}^{N} g(m)\right)^2\right\}^{1/2} \tag{8}$$

where

$g(*)$	are sample values of a component over an integral number of periods
N	is the number of samples over an integral number of periods
A_g	is the computed amplitude for component g(*).

Note the subtraction of the average value before squaring. Two running sums must be maintained, one for the average value and the other for the sum of squares.

An expression for relative phase is derived by expanding the time average value:

$$\begin{aligned}
&\left\langle [A\sin(2\pi ft + \phi) + B]\sin(2\pi ft)\right\rangle_{AVG} = \\
&\quad = \left\langle A\sin(2\pi ft + \phi)\sin(2\pi ft)\right\rangle_{AVG} + \left\langle B\sin(2\pi ft)\right\rangle_{AVG} \\
&\quad = \left\langle A(\sin^2(2\pi ft)\cos(\phi) + \cos(2\pi ft)\sin(\phi)\sin(2\pi ft)\right\rangle_{AVG} + \\
&\qquad + \left\langle B\sin(2\pi ft)\right\rangle_{AVG} \\
&\quad = A\cos(\phi)\left\langle \sin^2(2\pi ft)\right\rangle_{AVG} + \frac{A}{2}\sin(\phi)\left\langle \sin(4\pi ft)\right\rangle_{AVG} \\
&\qquad + \left\langle B\sin(2\pi ft)\right\rangle_{AVG}
\end{aligned} \tag{9}$$

Performing the time average over an integral number of periods yields:

$$\begin{aligned}
\left\langle \sin(2\pi ft)\right\rangle_{AVG} &= 0 \\
\left\langle \sin^2(2\pi ft)\right\rangle_{AVG} &= \frac{1}{2} \\
\left\langle \sin(4\pi ft)\right\rangle_{AVG} &= 0
\end{aligned} \tag{10}$$

Then collecting these results and solving for $\cos(\phi)$:

$$\cos(\phi) = \frac{2}{A}\left\langle [A\sin(2\pi ft + \phi) + B]\sin(2\pi ft)\right\rangle_{AVG} \tag{11}$$

The right-hand side can be computed from time sampled data:

$$\left\langle [A\sin(2\pi ft+\phi)+B]\sin(2\pi ft)\right\rangle_{AVG} = \frac{1}{N}\sum_{n=1}^{N} g(n)\sin(2\pi f(n-1)\Delta t) \tag{12}$$

Summation extends over an integral number of wavelengths. Note that this calculation is insensitive to any bias present. Equation (11) does not determine ϕ uniquely, however. Resolving the ambiguity is possible by comparing the first sample value against the bias. Finally:

$$\phi = \begin{cases} \cos^{-1}\left(2/AN\sum_{n=1}^{N} g(n)\sin(2\pi ft(n-1)\Delta t)\right), \\ \qquad\qquad g(1) < \frac{1}{N}\sum_{m=1}^{N} g(m); \\ 2\pi - \cos^{-1}\left(2/AN\sum_{n=1}^{N} g(n)\sin(2\pi(n-1)\Delta t)\right), \\ \qquad\qquad g(1) \geq \frac{1}{N}\sum_{m=1}^{N} g(m). \end{cases} \tag{13}$$

Using this technique, only two additional words must be reserved per field component to compute phase, one needed to accumulate the sum of (12) and the other to save $g(1)$.

In summary, an algorithm is available for computing the magnitude and relative phase for each field component that requires only four floating point words of storage per field component. The technique is insensitive to computational bias and becomes less sensitive to high-frequency computational noise as the computing window is increased without increasing the storage requirements.

8.2 The Connection Machine

a. Architecture

The CM is a massively parallel SIMD computer manufactured by Thinking Machines Corporation in Cambridge Massachusetts. The

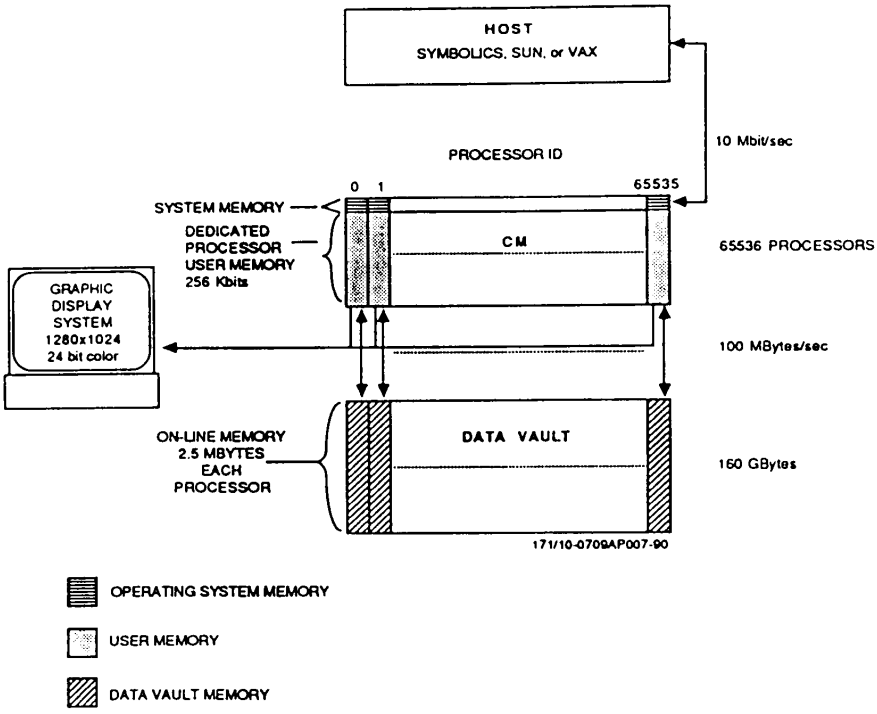

Figure 8.8 The Connection Machine.

work described in this chapter was performed during the 1987-1989 interval. Accordingly, the CM-2 used and described here was a "snapshot" of a new, evolving computer architecture. An excellent reference for the current CM-2 is [7]. A CM requires a host computer, either a SYMBOLICS LISP machine, a SUN workstation, or a VAX. The host supports serial computations and broadcasts instructions to the CM. The same instruction gets executed by each processor on data in its own memory. Figure 8.8 presents a pictoral sketch.

The high-level languages available are "parallel" extensions to C, LISP and FORTRAN. A set of macros called the Parallel Instruction Set (PARIS) is also supported as extensions to both LISP and C. LISP-PARIS was the preferred language for developing FD-TD software on the CM because it provided maximum flexibility to the user for man-

aging memory. Problem size, runtime and high-speed visualization are the three main reasons why users would be interested in using the CM to solve electromagnetics problems. Consequently, using the available RAM at its maximum efficiency was the determining factor for language selection. An excellent LISP-PARIS reference is [8].

Connection Machine systems are commercially available with 65536 processors each sitting on 262144 bits of dedicated RAM memory and a total of 160 GBytes of on-line disk storage called a data vault (DV). On-line storage is not shared memory and is best described as extending the dedicated memory on each processor by an additional 2.5Mbytes. Instructions are bit oriented giving the programmer the unusual, but extremely useful, flexibility to match word length to suit desired computational dynamic range and to match interprocessor message lengths to the actual data transfer needs. Floating point coprocessors are available supporting both 32 and 64 bit arithmetic calculations. An elapsed time for a complete cycle of single precision arithmetic (i.e., retrieve operands, perform the floating point arithmetic and store the result) of $40\,\mu$ sec is available using non-optimized non-pipelined code. Assuming that all 64K processors are productively computing, 1.6 Gigaflops ($65536/40\,\mu$ sec) establishes a measure of machine capability. Comparing FLOP† rates between serial and parallel machines can be misleading because different algorithms are often used on the architectures to solve the same problem. A more meaningful measure is elapsed time to solve the same problem. General processor-to-processor communication is available through a reduced hypercube connectivity network; however, for the algorithm capabilities incorporated into the present model, only nearest neighbor communication is used. For a 32-bit message the elapsed time ranges from $30\,\mu$ sec to $140\,\mu$ sec depending on algorithm design implying that a net memory transfer rate of at least 1.87 GBytes/sec is possible on a full machine.

The host computer plays an integral part in executing algorithms. Not only does it broadcast instructions to the CM, but it can also read data out of individual processors, perform computations on it, and write it back to a set of processors. This affords the user with flexibility to use both serial and parallel computational capability as needed. Programs can be written on small (4K) machines and can be run on larger machines without changing the code. This is accomplished by exploiting the operating system software supplied machine constants.

† FLOP is a commonly used acronym for floating point operation.

The concept of virtual processors is supported by the system software. Each physical processor can partition its memory by factors of 2 and assign to each partition a virtual processor ID or cube-address. An 8K machine can therefore act like a 64K machine provided that each virtual processor needs only 1/8 of a processor's physical memory, but at a cost of running 8 times slower and allocating stack space for each virtual processor when only one allocation is actually needed. The user is also able to manage the virtualization whenever it becomes beneficial to virtualize by factors other than powers of 2. This capability was exploited for FD-TD. Each processor was virtualized 60 times. Although the burden required to manage the virtualization is greater, the benefits are impressive. System-supplied software would support only 32 virtualizations. Furthermore, managing your own memory allows a user to assign some shared memory to each physical processor including stack space which supports the virtual processors assigned to the physical processor. Consequently, constants common to virtual processors need to be stored only once. Physical properties associated with multilayered scatterers must be stored only once per physical processor and indexed by virtual processors as needed.

b. Programming Philosophy

There are three broad classes of computer architectures that are generally recognized: serial, Multiple Instruction Multiple Data (MIMD), and Single Instruction Multiple Data (SIMD). Although the CRAY series of computers are often viewed as serial machines; in fact, they really are MIMD ones with a large shared memory. Most electromagneticists are probably familiar with at least one of these architectures. With the ever pervasive PC, the notion of a serial algorithm is so dominant in our thinking that it often influences our approach to solving analytical problems as well, frequently to our detriment. What has been surprising to us is the rich content of parallelism in seemingly serial algorithms. Algorithms that do not use communications are often inferior in speed to those using communications, because a reduction of serial algorithm order is possible on parallel architectures when interprocessor communications is exploited. The decreased runtime due to the reduction in algorithm order often compensates for slow interprocessor communications.

Although parallel computing is still in its infancy, there appear to be a few common characteristics among better programs. Segregating

communication requirements from computational requirements is one of them. SIMD computing efficiency falls whenever special cases must be addressed. Rather than executing the special cases serially, a more efficient approach is to aggregate the communications requirements among the cases and execute them together. Some computational requirements can be aggregated in a similar fashion. In this way the case specific operations are minimized, keeping the productive utilization of the processors high, and the slower communication operations more efficient because they will be smaller in number with messages longer in length.

Another good programming practice is to perform processor context manipulation in the lowest level functions possible. Every processor has a one-bit context flag. Processors can be turned on or off by toggling this flag. By setting context for a selected processor subgroup code execution can be restricted to the selected processor subset. Suppose a number such as:

$$\sin\left(2\pi\frac{\text{cube-address}}{65536}\right) \tag{14}$$

is to be added to a variable in a specific subset of processors. Rather than manipulating context in a high-level function to identify the processor subset, a better way is to create a function which initially sets cube-address (or processor ID) to zero, copies the correct cube-address into the desired subset of processors, and computes expression (14). This function can now be called from a higher level software function without context manipulations, simplifying debugging and checkout. Processors not intended to participate simply add zero.

8.3 Mapping FD-TD Onto a Connection Machine

a. Objectives

The CM code was developed under the following guiding principles in order of priority:

1. Maintain the computational accuracy available in the basic algorithm.
2. Maximize the size of the scatterer that can fit into the volumetric grid.
3. Minimize the runtime.

Computational accuracy is obviously important; in fact, FD-TD analysis of 10 λ flat PEC plates can exhibit monostatic backscatter variation of over 50dB with only a 16° change in incident direction. Remember that the ultimate utility of a scattering code lies with its predictive capability. Currently, our algorithm is running with single precision (32-bit floating point arithmetic), which has been shown to provide 50dB of dynamic range. With a little additional effort, the program can be converted to double precision, but double precision requires twice the memory reducing the maximum body volume by a factor of two. A more computationally efficient approach that would maintain the present scatterer size is to use a fourth order differencing scheme and single precision arithmetic. Scatterer size is favored over runtime because many scatterers of interest stress contemporary memory resources. A high-speed code solving uninteresting small problems is not useful.

b. Algorithm Layout

A close inspection of the fundamental FD-TD algorithm based upon the Yee Lattice and Mur boundary conditions [2] yields a decomposition into necessarily serial and optionally parallel processing steps. A first order breakdown of the algorithm specific serial processing steps for each time step is:

Update incident fields
Update grid interior H fields
Update grid interior E fields
Update terminating boundary plane edge E fields
Update terminating boundary plane interior E fields

With a staggered computational grid for E and H, it is possible to terminate at the boundary all grid calculations with just one of these vector fields leaving the other to be updated completely using Maxwell. Following Taflove, E field termination was employed. There is complete grid symmetry with this approach, a real advantage for debugging and noise floor assessment.

Efficient utilization of a parallel machine occurs when all processors contribute to the solution of the problem. The FD-TD code, as implemented here, permits this efficient processor utilization only if at least 2 dimensions of the problem space is as large as the number of physical processors. For example, an 8K machine (2^{13} processors)

Computational accuracy is obviously important; in fact, FD-TD analysis of 10 λ flat PEC plates can exhibit monostatic backscatter variation of over 50dB with only a 16° change in incident direction. Remember that the ultimate utility of a scattering code lies with its predictive capability. Currently, our algorithm is running with single precision (32-bit floating point arithmetic), which has been shown to provide 50dB of dynamic range. With a little additional effort, the program can be converted to double precision, but double precision requires twice the memory reducing the maximum body volume by a factor of two. A more computationally efficient approach that would maintain the present scatterer size is to use a fourth order differencing scheme and single precision arithmetic. Scatterer size is favored over runtime because many scatterers of interest stress contemporary memory resources. A high-speed code solving uninteresting small problems is not useful.

b. Algorithm Layout

A close inspection of the fundamental FD-TD algorithm based upon the Yee Lattice and Mur boundary conditions [2] yields a decomposition into necessarily serial and optionally parallel processing steps. A first order breakdown of the algorithm specific serial processing steps for each time step is:

Update incident fields
Update grid interior H fields
Update grid interior E fields
Update terminating boundary plane edge E fields
Update terminating boundary plane interior E fields

With a staggered computational grid for E and H, it is possible to terminate at the boundary all grid calculations with just one of these vector fields leaving the other to be updated completely using Maxwell. Following Taflove, E field termination was employed. There is complete grid symmetry with this approach, a real advantage for debugging and noise floor assessment.

Efficient utilization of a parallel machine occurs when all processors contribute to the solution of the problem. The FD-TD code, as implemented here, permits this efficient processor utilization only if at least 2 dimensions of the problem space is as large as the number of physical processors. For example, an 8K machine (2^{13} processors)

can efficiently solve Maxwell's equations in a volume corresponding to $2^n \times 2^m \times z$ cells as long as $n + m = 13$ and sufficient memory is available in each processor to support z cells. The physical size of the space, of course, depends on the size of the cell. Cell sizes of $\lambda/10$ and $\lambda/20$ are typical.

Each computational step above depends on the results of the previous one. Unless the basic algorithm is changed, these steps are sequential on both serial and parallel architectures. All spatial points to which the calculation in each step applies, can be updated in parallel, theoretically. In the second serial processing step, for example, all three components of H can be updated simultaneously, but in the current version of the code field components are updated sequentially. This choice favored analysis of larger scatterers at the expense of runtime as laid out in the guiding principles.

Every virtual processor was assigned the role of updating each of the six field components assigned to an FD-TD cell. A pictoral view appears in Fig. 8.9.

The cell anchor point is labelled and plays the role of origin for referencing cell field components. Numerically, the anchor point is the virtual processor cube address. Figure 8.10 presents a global perspective of the entire computational grid.

After the time counter gets incremented, all cells in an xy-plane get updated in parallel. The algorithm steps serially through the z-planes until the entire grid gets updated. Each CM physical processor maintains a stack of FD-TD cells by indexing virtual processors as z-planes. The i and j indices in Taflove's equations [2] index over the CM processors viewed as a two-dimensional plane. The index k steps through memory in each processor. From another viewpoint, indexing over k is like indexing through an array dimension on a conventional serial architecture.

c. Data Layout

Careful memory management is needed in order to maximize the size of the scatterer that can fit into CM RAM. Memory requirements are categorized as either shared or dedicated. Memory allocated to each virtual processor is dedicated memory and represents the memory required to support each FD-TD cell update for the duration of the run. Shared memory is memory allocated to each physical processor that is shared by all virtual processors that a physical processor supports.

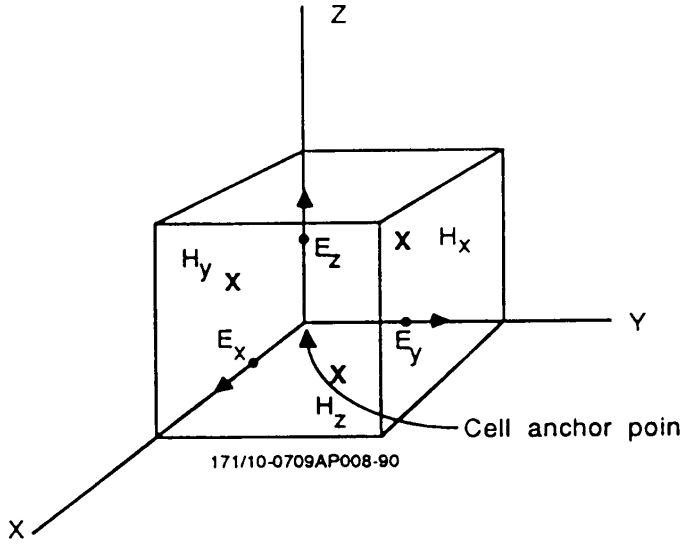

Figure 8.9 Virtual processor field assignment.

Problem size is also driven by the modeling generality required. For example, a code that supports computations on anisotropic scatterers requires more memory than that required to service isotropic ones. Host storage requirements are not significant.

The number of grid z-planes or number of virtual processors that each physical processor can support is calculated as follows:

$$\text{NVP} = \frac{\text{PPM} - \text{SPM} - \text{PSS} - \text{SSS}}{\text{VPM}} \tag{15}$$

where

NVP	is the number of grid z-planes
PPM	is physical processor memory
SPM	is shared memory per physical processor
PSS	is program stack space
SSS	is system stack space
VPM	is virtual processor dedicated memory

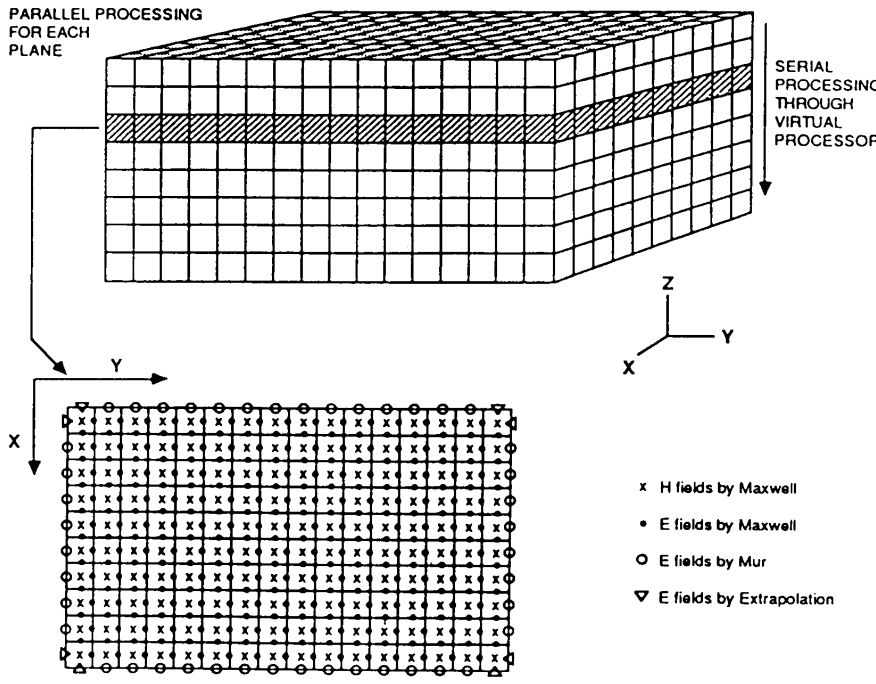

Figure 8.10 FD-TD computational layout.

Physical processor memory is strictly a function of the hardware configuration and varies from 65536 bits to 2.5 Mbytes depending upon RAM memory chip selection and data vault size. In this subsection sizing will be limited to RAM. The following section will discuss extension to the data vault. System stack space requirements are on the order of 1500 bits and program stack space is approximately 1200 bits for FD-TD.

Each of the algorithm specific serial processing steps enumerated above in the algorithm layout have different memory requirements. For example, Mur boundary updates need two time step old E field data; interior field updates do not. Magnitude and phase storage needs to be allocated, but not for the fields in the terminating grid cells. Table 8.1 identifies the virtual processor dedicated memory requirements for

Memory Word Index	FD-TD Cell Type		
	Edge	Mur	Grid Interior
1	ex	ex	ex
2	ey	ey	ey
3	ez	ex	ez
4	ex-1	ex-1	ex-mag
5	ey-1	ey-1	ey-mag
6	ez-1	ez-1	ez-mag
7	ex-2	ex-2	hx-mag
8	ey-2	ey-2	hy-mag
9	ez-2	ez-2	hz-mag
10	hx	hx	hx
11	hy	hy	hy
12	hz	hz	hz
13			ex-ph
14			ey-ph
15			ez-ph
16			hx-ph
17			hy-ph
18			hz-ph
19			ex-sum
20			ey-sum
21			ez-sum
22			hx-sum
23			hy-sum
24			hz-sum
25			ex-gl
26			ey-gl
27			ez-gl
28			hx-gl
29			hy-gl
30			hz-gl

Table 8.1 FD-TD memory map for each virtual processor.

Mnemonic	Descriptor
ex, ey, ez	E field component
hx, hy, hz	H field component
component-1	One time step old component value.
component-2	Two time step old component value.
component-mag	Magnitude of component. Also used as temporary storage.
component-ph	Phase of component. Also used as temporary storage.
component-sum	Accumulation of component sum over integer number of cycles.
component-gl	First value of component during magnitude and phase data accumulation period

Table 8.2 Memory word map key.

each cell type. Field overlaps are used whenever possible. Magnitude and phase memory is used for accumulating products and field component squared partial sums and for saving the final values until far-field quantities get calculated. Table 8.2 defines the mnemonics appearing in Table 8.1.

Each virtual processor must reserve the storage requirements summarized in Table 8.1 for the entire solution period. Note that the grid interior cells determine storage requirements, 30 floating point words. Using single precision arithmetic (32 bits), Virtual Processor Memory (VPM) is 960 bits.

The current version of the code uses 3677 bits of shared memory per processor and was sized to service a maximum of 4 material layers. Each layer represents an electromagnetic material with uniquely specialized ϵ_R, μ_R, σ, ρ. Approximately 100 bits of this total is used for flags. The remaining allocated memory space is used to store coefficients needed to update Maxwell's equations. Unique coefficients

Number of Processors	Memory per Processor (bits)	Volumetric Grid Size (Cells)	Volumetric Grid Size in Wavelengths ($\lambda/20$ cell size)
64K	64K	$254 \times 254 \times 60$	$12.7 \times 12.7 \times 3$
64K	256K	$254 \times 254 \times 264$	$12.7 \times 12.7 \times 13.2$

Table 8.3 FD-TD problem size for an in memory code.

are needed for each layer. As the z-planes get processed, the proper coefficients are indexed and moved from shared memory to stack for calculations.

Sufficient information is now available to compute the number of virtual processors (z-planes) in the grid. Results are summarized in Table 8.3 for the two RAM chips that are available. Although a 64K processor CM has a processor grid 256×256, only a 254×254 cell grid can be supported. In the initial algorithm formulation, terminating planes coincide with tangential E fields. An even number of cells or an odd number of cell faces are maintained along each grid direction. As a result, a 255×255 processor grid is active. Our experience suggests that a $\lambda/20$ gridding or better is needed to accurately match experimental data for most structures of current interest. On a fully complemented CM a $10\lambda \times 10\lambda \times 10\lambda$ scatterer can easily fit into CM RAM.

d. Extending the Algorithm to Use the Data Vault

Out-of-memory codes are frequently more complex than in-memory codes because efficient use of on-line storage media often requires memory staging. Extending the CM in memory code to use the DV is particularly simple when the DV is used to extend the grid in only one dimension. An efficient technique to achieve this goal will now be described.

The key observation is to view the DV as merely extended memory to each processor. The DV simply allows more z-planes to be added to the volumetric grid. Maximum grid cross-section is determined by

the number of CM processors. On a 64K CM, the grid cross-section is limited to 254 cells by 254 cells or for a $\lambda/20$ gridding $12.7\,\lambda$ by $12.7\,\lambda$.

The number of z-planes can be calculated by computing the RAM per processor including the DV and applying equation (15). The extended memory size is:

$$\frac{160 \text{ GBytes}}{65536} \times \frac{8 \text{ bits}}{\text{byte}} = 262144 \text{ bits/proc.} = 19.8 \text{ Mbits/proc.}$$

Substituting this memory value for PPM in (15) yields 20,610 z-planes. At $\lambda/20$ the grid z dimension in wavelengths now becomes 1030.5λ and the total volumetric grid resolved at $\lambda/20$ can be extended to:

$$12.7\lambda \times 12.7\lambda \times 1030.5\lambda\,.$$

Leaving a 1λ scattered field computational region around the scatterer implies that a scatterer as large as:

$$10.7\lambda \times 10.7\lambda \times 1028.5\lambda$$

is analyzable.

Interesting scatterers are not necessarily long and thin. The easiest way to redress the situation is to virtualize the CM by powers of two using the out of memory code just described. Virtualizing by small factors like two or four does not waste too much memory. The only memory that gets duplicated needlessly is the shared memory per processor, approximately 4000 bits, and the program stack space, 1200 bits. Virtualizing both the x and y dimensions by a factor of 4 will yield a total volumetric grid resolved at $\lambda/20$:

$$50.8\lambda \times 50.8\lambda \times 64.4\lambda$$

and if a 1λ scattered field computational region is reserved then a scatterer as large as:

$$48.8\lambda \times 48.8\lambda \times 62.4\lambda$$

will be analyzable. If $\lambda/10$ resolution provides sufficient accuracy, then the scatterer size could be as large as:

$$97.6\lambda \times 97.6\lambda \times 124.8\lambda$$

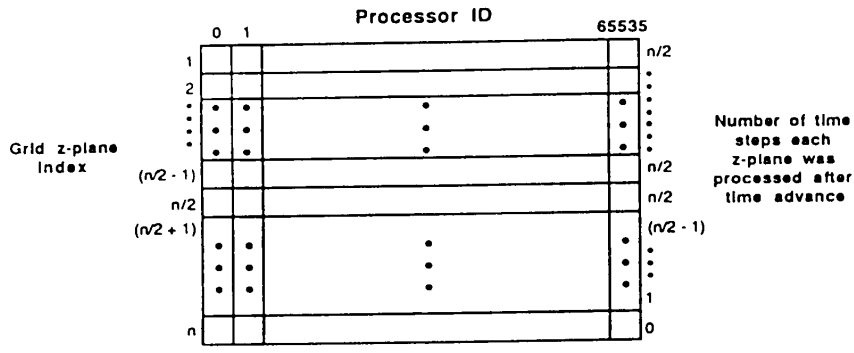

Figure 8.11 First load of data in CM after advance of $n/2$ time steps.

on hardware that is commercially available today.

The number of DV reads and writes can be minimized by processing the data in the CM as far ahead in time as possible. This concept becomes clear with the help of some Figures, 8.11 through 8.13. Recall that second-order differences require only nearest neighbor data. Since the entire xy-plane fits into the CM, the only nearest neighbors that require further consideration are z-plane neighbors. An algorithm for advancing the entire grid by $n/2$ time steps is now described. Figure 8.11 shows the update status for the first n z-planes that have been loaded into CM memory. The number of time steps that each z-plane was processed is labelled down the right-hand side. The first $(n/2)$ z-planes can be advanced $(n/2)$ time steps. Note that the first plane can be advanced because it is a grid termination plane. Obviously the last z-plane in memory cannot be advanced at all because there is insufficient data to update it. The one just above can be advanced only one time step for similar reasons. Processing continues by reading out the first $(n/2 - 1)$ z-planes, each having been advanced $(n/2)$ time steps into the future. The bottom $(n/2 + 1)$ z-planes are moved up and the next set of $(n/2 - 1)$ z-planes are read in below. Figure 8.12 captures this event. Time step advancing starts over for this memory load. Time is advanced until the status depicted in Fig. 8.13 is reached,

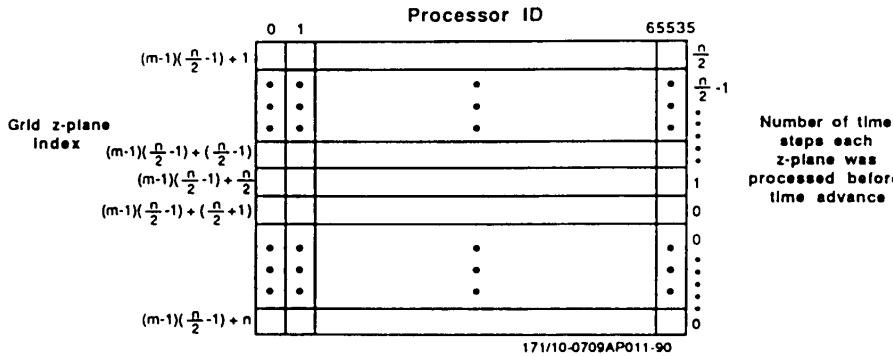

Figure 8.12 CM Memory Status with data for mth load into CM before time advancing.

the same time status that was achieved after processing the first load only now it has been achieved for the mth load.

Processing continues until the entire grid gets updated $n/2$ time steps. Then, the process starts over beginning with the first DV load and again each DV load is advanced $n/2$ time steps. The cycle gets repeated until steady-state is achieved.

A brute force approach would be to advance one time step and cycle through all DV loads until the entire grid is updated for that time step. For a volumetric grid having N z-planes the number of DV loads would be:

$$\text{Number of loads} = \left[\frac{N}{n-2} + 0.5\right]_{INT} \times M \tag{16}$$

where

N is number of z-planes in the volumetric grid
n is the number of z-planes that fit into one CM load
M is the number of time steps required to achieve a solution.

In equation (16) N is divided by $(n-2)$ because the last z-plane in memory cannot be updated in the current memory load. An

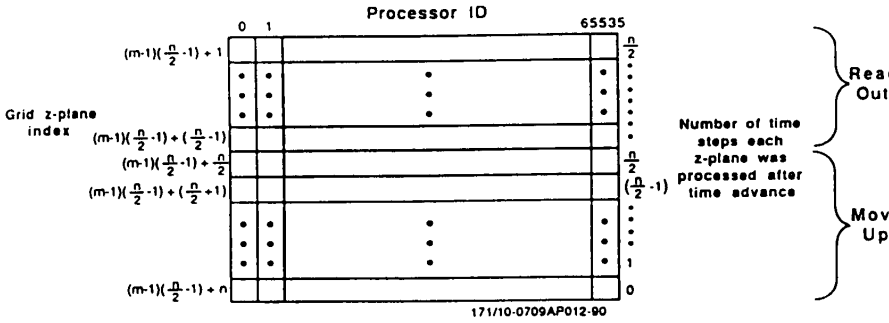

Figure 8.13 CM memory status with data for mth load into CM after time advancing.

expression can be easily derived for the required number of loads from the DV for the new algorithm:

$$\text{Number of Loads} = \left[\frac{N}{n/2-1} + 0.5\right]_{INT} \times \frac{2M}{n} \tag{17}$$

where

N is number of z-planes in the volumetric grid
n is the number of z-planes that fit into one CM load
M is the number of time steps required to achieve a solution.

Taking the ratio between equations (16) and (17) provides a comparison:

$$\text{Number of Loads Ratio} = \frac{n\left[\frac{N}{n-2} + 0.5\right]_{INT}}{2\left[\frac{N}{n/2-1} + 0.5\right]_{INT}} \tag{18}$$

The savings in DV loads is $0(n/4)$. On a CM equipped with 256Kbit memories n is approximately 200. The number of DV loads can be reduced by a factor of 50.

e. Using the Graphic Display System

The graphic display system is a high-speed 24-bit color 1280×1024 pixel display connected directly to the backplane of the CM. Read out data rates on the order of 30 MBytes/sec are achievable. The easiest quantities to display are the ones currently being calculated such as instantaneous field values, field magnitudes, and relative phases. The field plots attached to the example problems in Section 8.6 were taken from the graphic display system. A SEIKO printer was connected into the RGB output of the graphic display unit to obtain hardcopies.

Having a capability to plot field values in real time is an extremely valuable debugging tool, not only during code development but also for geometry input. Since fields inside PECs are zero, a traveling wave is easily discernable from the silhouette of the scatterer. There is a surprising amount of complexity visible in the instantaneous field patterns as fields attach themselves to surfaces and detach again at material discontinuities. Although our experiences to date are largely qualitative, we believe that research into field representation and display will produce tools that will assist both expert and novice designers to modify scatterer designs to achieve desired performance characteristics.

f. Remarks on Geometry Generation

Progress in the development of electromagnetic analysis software has been so rapid that within just a few years the bottleneck that designers will probably face is geometry generation. There are only a handful of software packages that are on the market today that appear to be suitable for geometry input, but they are written for specific workstations. As far as we know, no software is available for the Connection Machine.

8.4 Algorithm Testing and Checkout

a. Overview

There is a surprising amount of work needed to insure that the algorithm is working properly both from a computational standpoint and a physical one. Complete reliance cannot be placed on Maxwell's equations alone because the grid termination equations and computational interfaces do not arise from Maxwell's equations. At steady state,

however, the results of Maxwell's equations must obviously dominate at least down to the desired level of accuracy.

A surprising amount of insight can be gleaned from even a one-dimensional FD-TD model; the reader is strongly encouraged to construct one and learn some of the modeling limitations. The Mur boundary condition, for example, does not attenuate high frequency grid noise very well. In order to see this, introduce one incident wave cycle at one cell and examine the evolution to steady state. One-dimensional FD-TD problems can be run 10,000 or more time steps quickly.

FD-TD is a "subscript intensive" algorithm and debugging can be a laborious process independent of the computer architecture. There are no special algorithm instabilities or numerical noise sources associated with parallel processing. Two tests are described below that provided a wealth of debug information. They would be useful for testing algorithm development on any architecture.

b. Free Space Scatterer

In addition to providing debug on the total field scattered field calculation, running the code without a scatterer provides an estimate of the computational noise floor. Table 8.4 defines a small volumetric grid that was set up on an 8k processor CM. Figure 8.14 is a plot of the bistatic scatter. Theoretically, the dB level should be $-\infty$ for all angles. A less than ideal result occurs because noise is added at the total field scattered field interface. The noise floor is not random. Actually the computational grid is a cavity with very lossy walls. The noise floor cannot be viewed as additive noise when a scatterer is present because the presence of the scatterer disturbs the field distribution, i.e. there is mutual coupling between the scatterer and the computational noise sources.

On the CM it is very efficient to generate the incident field at all points on the total field scattered field interface in parallel using an analytical expression for a propagating sinusoid with the wave number corrected for the grid resolution. In his serial code, Taflove generates the incident wave using a one-dimensional FD-TD model and interpolates. For propagation along a coordinate axis, Taflove produces lower reflections than our analytical approach, but for other incident wave directions his noise floor could be higher. A controlled comparison would

Attribute	Description
Volumetric Grid ($\lambda/20$) (Wavelengths)	$6.1\lambda \times 3.1\lambda \times 1.5\lambda$
Total Field Computational Region (Wavelengths)	$5.3\lambda \times 2.3\lambda \times 0.7\lambda$

Table 8.4 Volumetric grid specifications for free space bistatic plot.

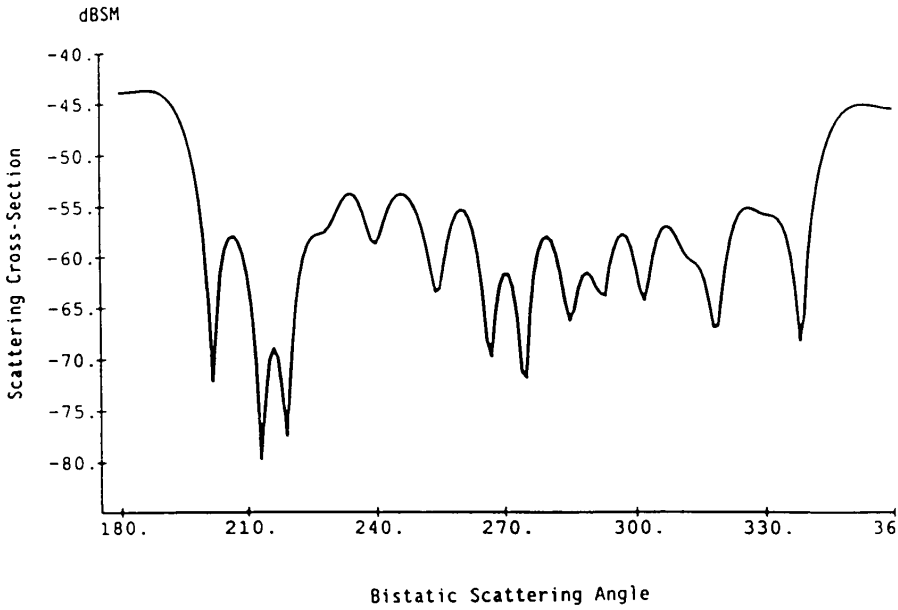

Figure 8.14 Bistatic plot for free space scatterer. Volumetric grid is a cavity with lossy walls.

require matching grid dimensions and computational boundaries. It will be conducted at a later date as part of an overall accuracy study for FD-TD.

c. *Dipole Sources*

Any source can be introduced into the grid. A dipole source is important because it facilitates designing antennas. The dipole source is also an excellent diagnostic source, because it enables checkout of the Maxwell update equations and the grid termination conditions. Introduction into the grid is easy. Simply overwrite one of the E field components with a sinusoid:

$$E_{ijk} = A \sin(2\pi f n \Delta t)$$

where

E	is dipole E field at cell (i,j,k)
A	is amplitude
f	is desired frequency
n	is the time step index
Δt	is the time step.

The time step Δt is constrained and must be chosen to satisfy the Courant stability condition. For a uniform grid Courant stability requires:

$$\Delta t < \frac{\delta}{\sqrt{3}\,c}$$

where

δ	is cell edge length
c	is speed of light.

Obviously, an E_x, E_y, or an E_z dipole can be introduced and is a function of the field component that gets overwritten. Maxwell's equations generate the H field necessary for wave propagation. Placing the dipole in the grid center sets up a totally symmetric situation where outgoing spherical waves get generated and get absorbed by the grid termination conditions.

d. *Performance*

Algorithm performance on a CM will be addressed from several perspectives. One technique for categorizing the state of code is to classify it as, a quick installation, a fine tuned version, or at peak performance. The present code can be classified as fine tuned using this measure. Comparisons between machines is difficult. One criterion might be elapsed runtime/problem size/dollar, but this approach

makes sense only if the machine is a dedicated resource to FD-TD. Our approach will not address any of these potential issues directly, but, instead, provide the reader with sufficient information on how well the CM architecture was/can be exploited when running FD-TD. Hopefully, it will be possible for the reader to compare performance measures for other architectures using this data.

An estimate of the CM MFLOP rate was made by benchmarking the runtime to perform 100 FLOPs: 25 single precision adds, subtracts, multiplies, and divides on an 8K machine hosted by a Symbolics 3670 running OS 4.3. The elapsed runtime was used to compute the MFLOP rate of an 8K CM using floating point 32-bit coprocessors. The result is 163 MFLOPS. Higher rates can be achieved by "pipelining"† but this capability is available only if the system software is used to manage virtualization. As explained earlier, larger scatterers can be analyzed if the user manages virtualization in the program. A more accurate estimate can be made if the FLOP breakdown into adds, subtracts, multiplies, and divides followed that in the software but the breakout wasn't readily available. The results would be different. Divides, for example, are twice as expensive as multiplies. All data was benchmarked on an 8K CM with floating point 32-bit coprocessors. Extrapolating to larger machines such as a 64K one is exact because all processors in a SIMD architecture execute the same instruction provided they are active.

The largest computational grid that will fit into an 8K CM with 64K-bit memories is:

$$62 \text{ cells} \times 126 \text{ cells} \times 60 \text{ cells}.$$

This grid contains over 2.8 million field unknowns. A program was written that counted the number of FLOPs executed by the program. Two values were produced, the total number of FLOP instructions broadcast by the host, and the number of FLOPs actually executed. The former represents 100% processor utilization while the latter weights by the actual number of processors being used. By comparing the 100% processor utilization figure against the peak, a measure of degradation due to nearest neighbor communications can be

† Pipelining for the CM means that partial sums and products are left in the coprocessor whenever the result is an operand for the next instruction and the next instruction is a CM floating point operation.

made. For this problem our performance benchmarked CM figure is 66 MFLOPS. Accounting for less than full processor utilization reduces the FLOP rate to 36 MFLOPS, the actual rate that is currently being realized. Estimating the improvement that can be expected from algorithm changes and a code rewrite suggests that the actual rate can be increased to at least 49 MFLOPS.

8.5 Illustrative Problems

Numerical analysis of electromagnetic systems provides both quantitative and qualitative information. Supercomputers applied to electromagnetic analysis and design provide more numerical data than can be reduced to useful quantitative information. Although quantitative data in conventional graphical formats will continue to be extracted from supercomputing systems, massive amounts of computed data can provide qualitative information essential to a researcher or a designer in a human-aided-design environment.

The examples on the following pages illustrate qualitatively the behavior of a three-dimensional, vector field in the presence of various materials and structures. More information on the behavior of fields, such as the direction of propagation, relatively high or low phase velocities, and cause and effect relationships can best be obtained by a time-stepped visual display, a movie, of a transient or sinusoidal wave interacting with the electromagnetic system under study. These movies are gaining popularity, not only because of their novelty, but because of the insight that they immediately provide to an electromagnetics researcher viewing wave behavior in both simple and complex systems. What remains to be accomplished is a practical assessment of the value of field visualization in the numerical design of complex, sensitive, electromagnetic systems.

a. Coated Surfaces

Figures 8.15 and 8.16 illustrate two examples of an FD-TD analysis of lossy dielectric and magnetic materials. The analyses were performed on material with ideal impedance-matched characteristics in order to ascertain the value of these materials. Such numerical analyses can preclude many costly "trial and error" experiments and can increase the usefulness of more productive experiments. Moreover, the field dis-

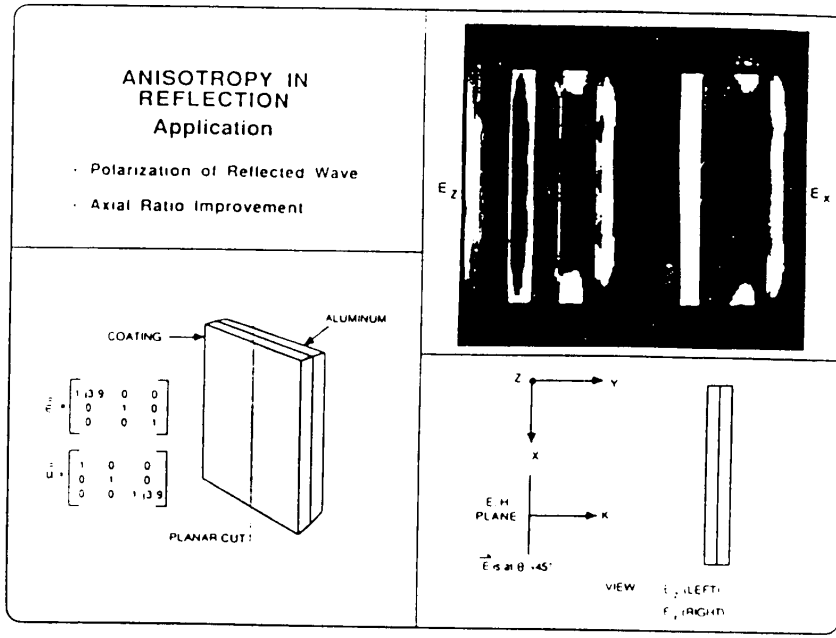

Figure 8.15 **The visual display interface of the CM and a Red-Green-Blue (RGB) hardcopy device provided the color picture in the upper right. The E_z and E_x fields (left and right respectively) of a single z plane in an 8192 processor CM were used to produce the pictures. Each view shows two computational regions; an inner "total field" region, and an outer "scattered field" region. Color scale is ROYGBIV with red representing peak positive fields; green-zero fields; and violet-peak negative fields. The example illustrates the selective absorption of a specific vector component E_x of a wave incident on a $4\lambda \times 3\lambda$ thin coated plate.**

plays were used to provide a better understanding of electromagnetic phenomena. In the case of Fig. 8.16, for instance, the completion and continuity of the phase front over the material coating on the edge of the plate provides new insight into wave behavior for such configurations.

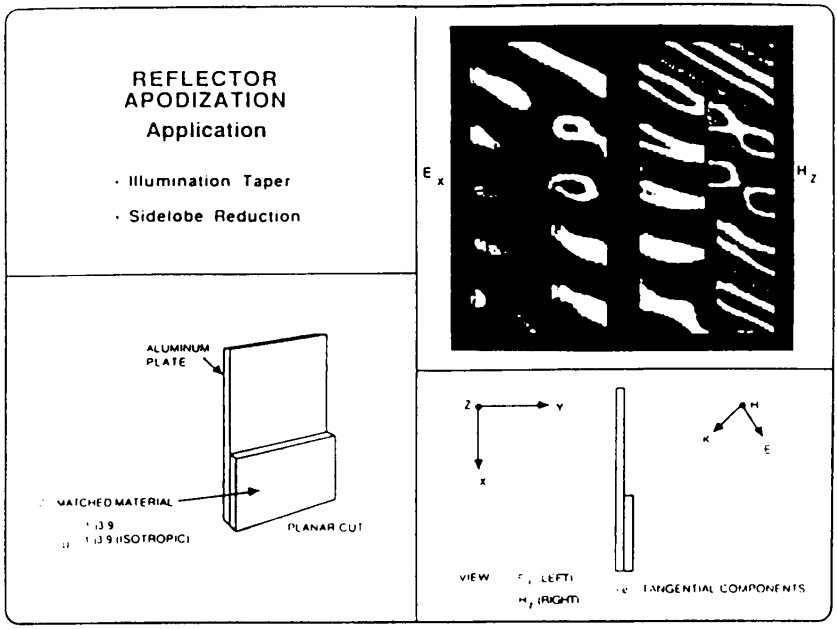

Figure 8.16 The visual display interface of the CM permits real time access to computed data. The visual display permits a researcher to observe phenomena that would be otherwise lost in a mass of graphs and numbers. The example illustrates the effect of impedance-matched, lossy material on the reflection of an obliquely incident wave. Only the total field region is shown surrounding the $4\lambda \times 3\lambda$ plate.

b. Waveguide Lens Antenna

Figures 8.17 and 8.18 illustrate the use of FD-TD to analyze a complex waveguide lens. Only through Maxwell equation techniques can the non-ideal behavior of a waveguide in the presence of other non-uniform waveguides be correctly analyzed. The analysis results shown in Fig. 8.18 were calculated along with all other field components at all points within the computational space-in approximately 20 minutes on a 16K CM. This problem, requiring 3.8 million unknown field components, is the largest solved problem reported in this chapter.

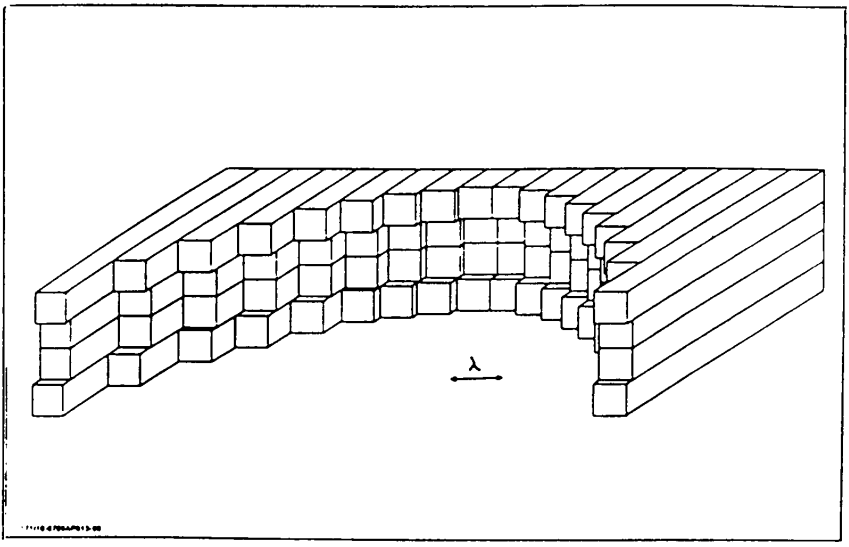

Figure 8.17 FD-TD calculations on a cartesian grid are naturally suited for modeling a waveguide lens. Here the lens was modeled with the "stairface" technique described in the text. The cross-section of each of the 72 waveguides is modeled by a 6×6 array of cells, which permits 216 unknowns (6 vector field components per cell) to be used to define the electromagnetic field over a cross-section of each waveguide. Maxwell's equations are used throughout the entire computational space.

8.6 Future Directions

The basic FD-TD algorithm is being extended to three-dimensional doubly curved convex PEC surfaces by distorting only those cell faces through which the surface passes. This extension is based upon Taflove's contour path interpretation [2]. Results are promising both from the standpoint of accuracy and runtime. They will be reported in another forum. These techniques apply, in principle, to concave scatterers also, but the geometry processing associated with locally concave regions is more complicated than that associated with convex surfaces. All of the major technical issues appear to have been addressed. Code development and validation of the modeling approach are still needed.

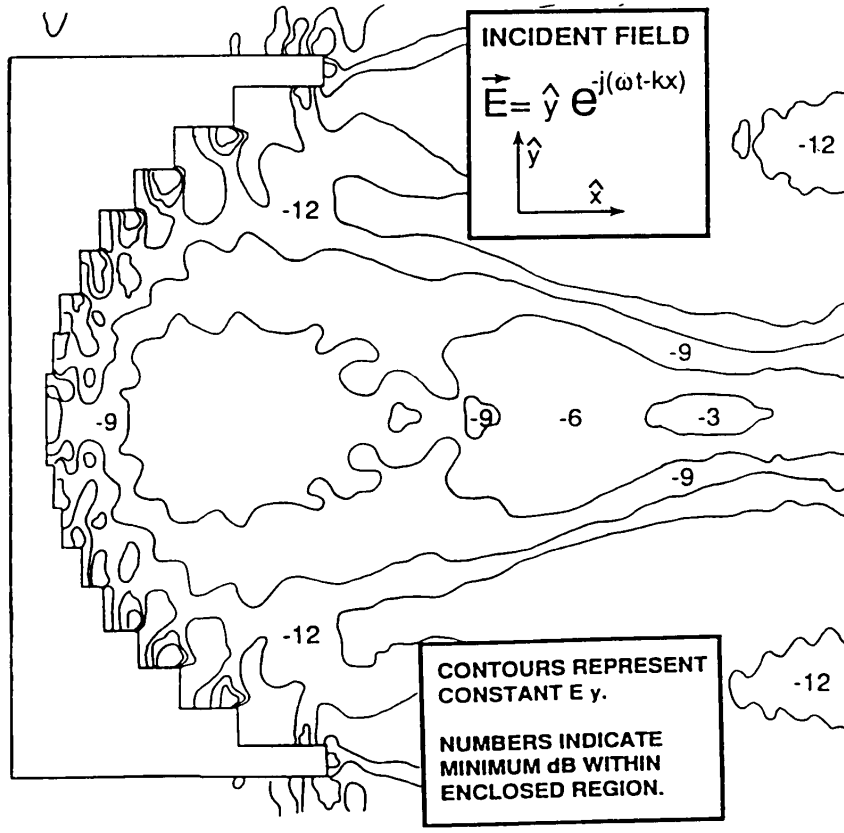

Figure 8.18 The steady fields in the focal region of the lens of Fig. 8.17 are shown for the case of an incident plane wave normal to the face of the lens. The steady-state fields were calculated from a time history of a single z plane of a 16384 processor CM at the completion of the 15th cycle. The FD-TD Maxwell equation solution of this lens captures the non-ideal behavior of the waveguides in the presence of other waveguides.

Acknowledgments

The authors wish to specially thank Dr. Taflove at Northwestern University and Dr. Korada Umashankar at the University of Illinois at Urbana for guidance. Their help enabled the MRJ electromagnetics group to grow and participate in advancing the forefront of compu-

tational electromagnetics in just a few short years. Dr. Taflove also provided the material for Section 8.2b. Tom Kraay, chief scientist at MRJ, is also acknowledged for his help and inspiration to develop fast and efficient parallel architecture algorithms. The authors thank Dr. George Wilson, Dr. Tor Opsahl, Mr. Paul Becker, Mr. John Hansen and Mr. Ruben Steck for their assistance. Ms. Billie Jo Dransfield prepared this publication.

References

[1] Taflove, A., and M. E. Brodwin, "Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell's equations," *IEEE Trans. Microwave Theory Tech.*, **MTT-23**, 623–630, August 1975.

[2] Taflove, A., and K. A. Umashankar, "The finite-difference time-domain method for numeric modeling of electromagnetic wave interactions with arbitrary structures," *PIER 2 Finite Element and Finite Difference Methods in Electromagnetic Scattering*, Elsevier, Chapter 8, 287–373, 1990.

[3] Weiland, T., "Numerical solution of Maxwell's equation for static, resonant and transient problems," *Proceedings of Union Radial Scientific International Symposium on Electromagnetic Theory,* Budapest Hungary, 25–29 August 1986.

[4] Weiland, T., "On the unique solution of Maxwellian eigenvalue problems in three dimensions," *Particle Accelerators*, **17**, 1985.

[5] Schelkunoff, S. A., "Field equivalence theorems," *Comm. Pure Appl. Math.*, **4**, 43–59, June 1951.

[6] Lee, S. W., and R. J. Marfehka (Editors), *Data Book of High-Frequency RCS*, Report by High Frequency Subcommittee Joint Services Electromagnetic Code Consortium Version 1, Urbana Illinois, July 20, 1989.

[7] *Connection Machine Model CM-2 Technical Summary*, Thinking Machines Corporation, Version 6.0, November 1990.

[8] *The Connection Machine System Paris Reference Manual*, Thinking Machines Corporation, Version 5.2, October 1989.