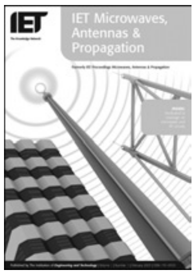
Published in IET Microwaves, Antennas & Propagation
Received on 20th May 2007
Revised on 9th October 2007
doi: 10.1049/iet-map:20070118



ISSN 1751-8725

Graphics hardware accelerated multiresolution time-domain technique: development, evaluation and applications

G.S. Baron[1] E. Fiume[1,2] C.D. Sarris[1]

[1]Department of Electrical and Computer Engineering, University of Toronto, Toronto, Ontario, Canada M5S 3G4
[2]Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G4
E-mail: cds@waves.utoronto.ca

Abstract: Recently, the use of graphics processing units as a means of achieving the hardware acceleration of the finite-difference time-domain (FDTD) technique has attracted significant interest in the computational electromagnetics community. However, the large memory requirements of the FDTD, compounded by the limited memory resources available in graphics processing units, compromise the efficiency of this approach. Alternatively, the authors show how the implementation of the multiresolution time-domain technique in a graphics processing unit can optimally utilise the memory resources of the latter and achieve unprecedented acceleration rates, significantly higher than those achievable by the FDTD. A detailed description of the proposed implementation is provided, followed by rigorous numerical error and performance evaluation studies that conclusively verify the advantages of the graphics accelerated multiresolution time domain. Finally, the potential of this technique as a fast microwave wireless channel modelling tool is demonstrated.

1 Introduction

The application of the finite-difference time-domain (FDTD) technique [1] to practical electromagnetic analysis and design problems is facilitated by its simplicity and versatility, but limited by the large computational resources needed for achieving convergent results, as a consequence of its pronounced numerical dispersion. Almost 10 years ago, a systematic way to formulate time-domain numerical methods with improved dispersion properties was presented with the introduction of the multiresolution time-domain technique [2]. Through homogeneous and inhomogeneous cavity numerical experiments, along with a Fourier dispersion analysis, [2] showed the potential of this technique to achieve relatively small dispersion errors, at coarse discretisation rates approaching the Nyquist limit of $\lambda/2$ (λ being the smallest simulated wavelength). On the other hand, although the use of multiresolution time domain allowed for the reduction in the number of cells within a given domain, it also led to increased operations per cell compared with the FDTD. As a result, the former has been understood to offer modest savings in simulation time compared with the latter. Still, a widely accepted advantage of the technique is its ability to yield accurate results even at coarse meshes, potentially reducing the number of required cells by almost two orders of magnitude in three-dimensional problems [2].

Within the computer graphics community, the area of general-purpose computing on computer graphics cards, formally referred to as graphics processing units, has emerged [3–5]. Originally designed for the specialised task of accelerating video games, modern graphics cards can also be employed for the execution of numerical operations, as long as the latter can be disguised as image processing transformations, such as shading or texturing. The idea of utilising graphics processing units for scientific computing has gained significant momentum, attracting wide research and commercial interest across multiple disciplines. In computational electromagnetics, this concept was introduced in [6, 7], where a two-dimensional FDTD scheme equipped with periodic and absorbing boundary conditions, was implemented, achieving acceleration rates of a factor close to 10, compared with the central processing unit implementation of the same technique. Larger acceleration rates were demonstrated when a uniaxial perfectly matched layer [8] set of equations was considered in [9], accompanied by an error analysis indicating that the numerical errors produced were within acceptable limits. Such a conclusion is reassuring for scientific computing applications, bearing in mind that graphics cards are intended to support visually convincing object rendering, rather than numerical accuracy. Also, Inman and Elsherbeni [10] introduced a high-level programming language implementation of the three-dimensional FDTD basic update equations (excluding boundary conditions) in a graphics card.

It is worth mentioning that the hardware acceleration of the FDTD has been successfully pursued in the past, using custom-made hardware [11–15]. What makes the case of graphics hardware-based scientific computing particularly attractive is that graphics cards are very fast, very cheap, and continue to be enriched at rates that continue to outstrip those of general purpose central processing units and field-programmable gate arrays. However, graphics hardware-based scientific computing is currently limited by the fixed amount of memory available in graphics cards.

This paper investigates the graphics card implementation of the scaling multiresolution time-domain technique of [2], extending previous work reported by the authors in [16]. Thus, the issue of the limited memory resources available in a graphics card is addressed by replacing the FDTD scheme with a much more memory efficient one. Furthermore, the numerical results of this paper suggest that the graphics processing unit performance actually improves with the arithmetic complexity of the programming involved. Therefore techniques invoking higher-order finite-difference approximations of spatial partial derivatives are shown to be ideally suited to graphics processor acceleration, precisely because they employ more arithmetic operations per cell. Hence, what has long been considered as a drawback of this technique becomes an advantage, as far as graphics processor acceleration is concerned, leading to unprecedented sustained acceleration rates, close to a factor of 30. Such rates are much greater than those previously reported for the FDTD.

The paper is organised as follows: Section 2 presents a brief overview of the scaling multiresolution time-domain field update equations. The concepts that were employed for the implementation of these update equations on a graphics processing unit are presented (for the first time in detail) in Section 3, where an analogy between time-domain simulations and image processing operations is outlined. In particular, the finite-difference equations describing a generic uniaxial perfectly matched layer medium are interpreted as discrete spatial convolutions that are routinely invoked in the functions performed by a graphics card. By virtue of this analogy, the card can be instructed to execute these update equations. Section 4 presents a detailed error characterisation analysis which conclusively verifies that the numerical imprecision apparent in the graphics card-based scaling multiresolution time-domain computations is practically negligible. The advantages of the proposed approach, compared with its FDTD counterpart, are most clearly demonstrated by jointly considering performance and accuracy of the two simulators. Such a study was performed for the first time and its results appear in Section 4. Finally, Section 5 demonstrates how this paper's technique can realise the simulation of electrically large problems that would not be solvable via a graphics accelerated FDTD implementation, because of memory limitations. To this end, a microwave indoor wireless channel time-domain simulation is shown to operate accurately two orders of magnitude faster on a graphics than a central processing unit. Section 5 summarises our contributions and alludes to future work.

2 Overview of the Deslauriers-Dubuc scaling multiresolution time-domain technique

The technique of [2] is formulated for a two-dimensional system of Maxwell's equations with respect to (H_x, E_y, H_z), employing Deslauriers–Dubuc bi-orthogonal scaling functions [17] for the expansion of field components in space. These basis functions are distinguished for being smooth, symmetric, compactly supported and exactly 'interpolating' (their value is zero at all non-zero integers). Hence, they facilitate the application of localised boundary conditions (such as perfect electric conductors) and the modelling of inhomogeneous media (such as perfectly matched layers).

According to the procedure outlined in [2], the field components are expanded in the Deslauriers–Dubuc basis functions ϕ in space and rectangular pulse functions h in time. For example, the E_y-expansion reads

$$E_y = \sum_{i,J,k} {}_kE_{i,j}\phi_i(x)\phi_j(z)h_k(t) \qquad (1)$$

with $\phi_p(\xi) = \phi(\xi/\Delta\xi - p)$, where $\Delta\xi$, $\xi = x,z$, is the cellsize in the ξ-direction and $h_k(t) = h(t/\Delta t - k)$, where Δt is the time step. The corresponding scaling multiresolution time-domain scheme is readily derived via the Galerkin method. Consider, for example, the y-component of Ampere's law

$$\epsilon\frac{\partial E_y}{\partial t} = \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} \qquad (2)$$

Following the notation of [2], the finite-difference form of (2), corresponding to the update equation of the electric-field component E_y at the node $(i\Delta x,\ j\Delta z)$ at the time step $n+1$, is

$$\begin{aligned} {}_{n+1}E^y_{i,j} &= {}_nE^y_{i,j} \\ &+ \frac{\Delta t}{\epsilon\Delta z}\sum_{p=-p_0}^{p_0-1}\alpha(p)_{n+1/2}H^x_{i,j+1/2} \\ &- \frac{\Delta t}{\epsilon\Delta x}\sum_{p=-p_0}^{p_0-1}\alpha(p)_{n+1/2}H^z_{i+1/2+p,j+1/2} \qquad (3)\end{aligned}$$

In (3), the spatial partial derivatives of (2) are approximated by finite differences, represented by weighted distributed sums. The weight coefficients $\alpha(p)$ are the so-called connection coefficients, given by the formula [2]

$$\alpha(p) = \frac{1}{2\pi}\int_{-\infty}^{+\infty}\lambda|\hat{\phi}(\lambda)|^2\sin(\lambda(p+0.5))\mathrm{d}\lambda \qquad (4)$$

Their numerical values are available in [18]. For these coefficients, $\alpha(-p) = -\alpha(p-1)$. The number of non-zero connection coefficients, $2p_0$, is related to the order of the Deslauriers-Dubuc function q, since $p_0 = 2q - 1$.

Another notable relation is that the Deslauriers–Dubuc functions of order $2k-1$ are produced by the autocorrelation of, and share identical connection coefficients $a(l)$ with, the Daubechies scaling basis functions of k vanishing moments [18]. As a result, the order of accuracy of their corresponding finite-difference operators [the distributed sums of (3)] is $2k$ [19]. This paper focuses on Deslauries–Dubuc functions of orders 3, 5 and 7 (depicted in Fig. 1) which result in higher-order spatial finite-difference operators (of orders 4, 6 and 8) compared with the second-order ones of the FDTD. The higher order of accuracy of the spatial finite differences produces the superior numerical dispersion properties featured by the scaling multiresolution time-domain.

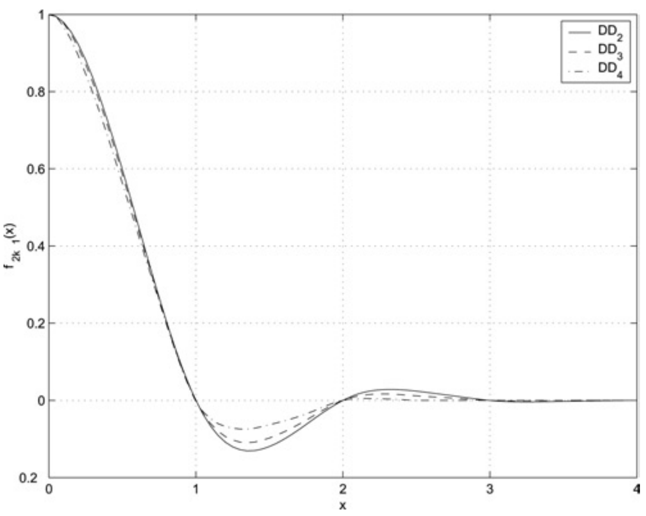


Figure 1 *Deslauriers-Dubuc interpolating basis functions of order 3, 5 and 7; DD_k denotes a Deslauriers-Debuc scaling function of order $2k-1$*

3 Scaling multiresolution time-domain technique as image processing and its graphics processing unit implementation

3.1 *Introduction*

Modern graphics processing units realise a so-called stream co-processor, a device that excels at computational problems involving large collections of data requiring complex, unordered and largely identical processing. These collections are termed streams and they are operated on by kernels. We call the application of a particular kernel to a stream a pass. Each pass involves its kernel being executed once for each input element, in one or more parallel pipelines, to produce a single corresponding output element.

On graphics cards, kernels are realised by so-called fragment shaders operating on input streams of pixel coordinates. These shaders support a floating-point vector instruction set [20] and have native constructs for spatially addressing two-and three-dimensional floating-point arrays (uniforms). The graphics terminology for those constructs and arrays are samplers and textures. Fragment shaders operate on every pixel of a rendered image, and their resulting output can be written back to the texture for reuse in subsequent passes. Fragment shaders as kernels are particularly well suited to two-and three-dimensional discrete spatial convolutions, heavily involved with the image processing operations carried out by the cards.

Given an input stream of texture coordinates, fragment shaders can sample corresponding and neighbouring pixels in read-only textures by appropriate off-setting.

In the following, our approach on how these inherent functional features of a graphics card can be employed towards the hardware implementation of a finite-difference method, such as the scaling multiresolution time-domain, is explained.

3.2 Time–domain simulations as state-space processes

Any linear time-invariant system of explicit difference equations can be interpreted in state-space form. The update equations governing time-domain simulations are no exception. In fact, the simulation's two-step leap-frog process, whereby a present state is determined from a linear combination of its two immediately preceding states [as in (3)], is characteristic of all explicit numerical solutions to second-order partial differential equations. The iterative application of this process is illustrated as a state-space simulation diagram in Fig. 2*a*. In this figure, the rectangular blocks depict state variables which characteristically constitute the nodes of the diagram. Circles alternatively represent state transition transfer functions and particularly linear operators on multiple inputs (e.g. collections of convolutions and their sums), required to affect only one state variable. By mapping electric and magnetic fields to the complementary grey and white blocks in Fig. 2*a*, their update equations are thus examples of such multi-input transfer functions. Such a simulation diagram captures the main idea behind the realisation of finite differences in a graphics card. This realisation entails defining an object for every state variable, and implementing a separate kernel program for each transfer function. The insight to implement update equations of the form of (3) as kernels is recognising them as discrete spatial convolutions. When disguised as such, they can be executed by the fragment shaders of a card.

3.3 Update equations as discrete spatial convolutions

Discrete spatial convolutions are independent linear operators on multi-dimensional arrays of previous state, and spatial refers to how these arrays are indexed. When applied, they independently determine a new value for every element by way of a weighted linear combination of neighbouring values. Given the canonical spatial indexing of field and material property meshes, the second and third term of (3) are discrete spatial convolutions weighted by connection coefficients, *a*(*l*). Even the first term can be interpreted as a convolution with an impulse, so that, in general, all update equations amount to sums of these convolutions. This concept is illustrated in Fig. 2*b*, which depicts the update of a magnetic field array (upper grey blocks) through the execution of convolution operations involving the array itself and an electric field array (white blocks). The output is the next state of the magnetic field array (lower grey blocks).

In our approach, we realise (3) by defining two-dimensional floating-point arrays Efield and Hfield for present electric and magnetic field states, and implementing a kernel to update the former while using the latter as a constant. In particular, these arrays would support a two-dimensional spatial addressing whereby the kernel would operate on an input stream comprising the unique addresses of individual electric field cells (that is, $\{(i,j)$, for all $i,\ j\}$). Upon conclusion, the resulting output stream would overwrite the Efield array in such a way to preserve that addressing for subsequent passes. A similar process is followed for the magnetic field update equations.

For the two-dimensional case, a very pertinent analogy with image processing emerges. Discrete spatial convolutions, and subsequently update equations, amount to image processing filters. The specific mapping of the update equations to graphics card functions involves the following three steps. First, we map the spatial dimensions of the simulation region to the dimensions of the textures processed by the cards. For our two-dimensional simulation region, we map the width of the image to the *x*-direction and the height of a two-dimensional texture to the z-direction, and define the lower left corner of the image as the origin.

The second step involves appropriately packing field values and material properties into separate textures and their colour channels. Textures in graphics are arrays of four-component colour values (white, black, grey and alpha or transparency). A summary of our six textures and their formats is illustrated in Fig. 3*a*. The values in each channel can be represented by a single-precision floating-point value. Design goals include minimising the memory footprint of images, and grouping values in a way that minimises the number of inputs used by the fragment shaders.

The final step is implementing fragment shaders for procedurally updating the Efield and Hfield textures. We implemented three shaders: UpdateE, SourceE, and UpdateH. In particular, an individual pair of UpdateE/UpdateH shaders were implemented for each of the Deslauriers–Dubuc-based schemes, whereas SourceE masks the Efield texture with a transparent source excitation [1]. Internal to these shaders, perfect

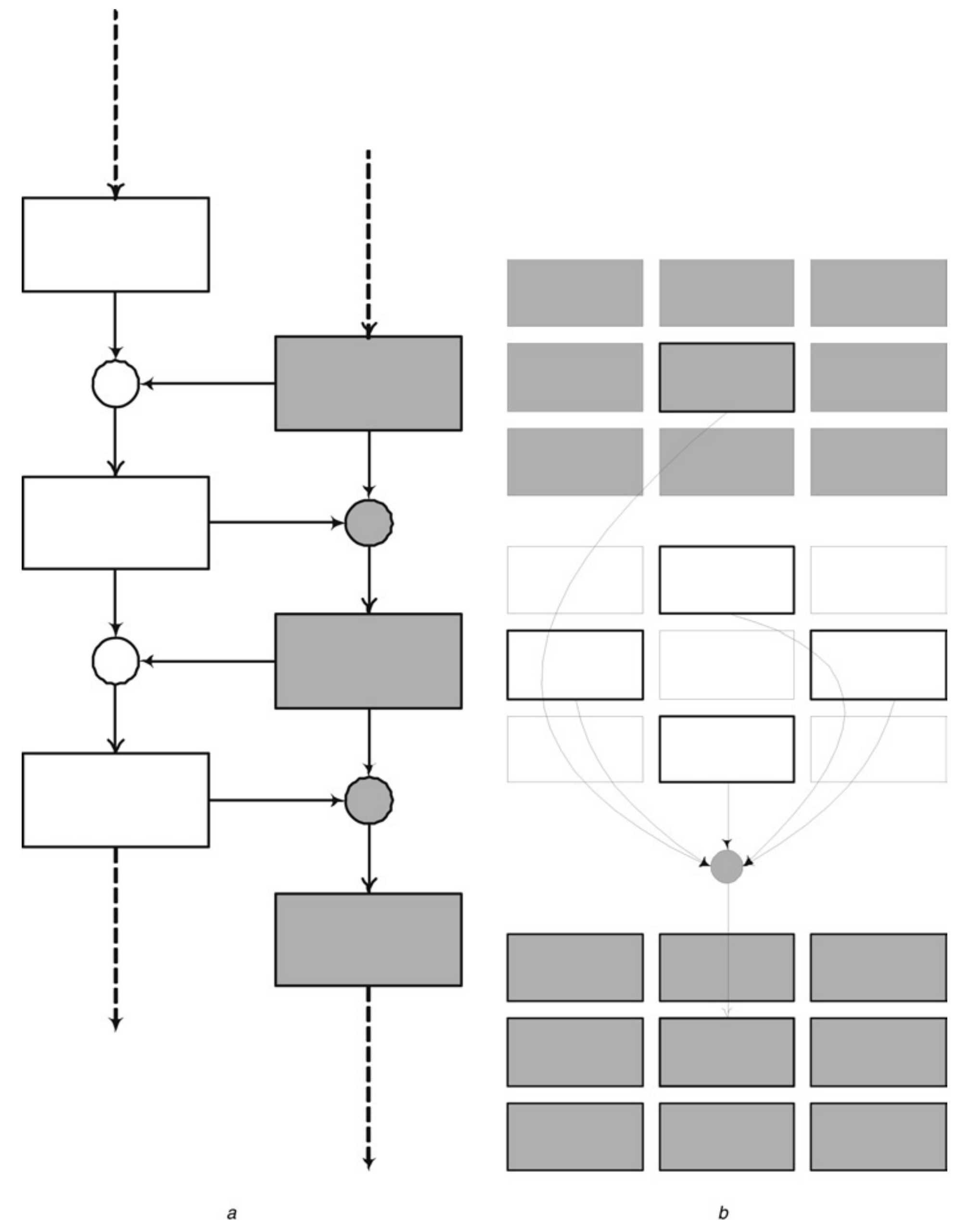


Figure 2 *Representation of operations involved with an electromagnetic time–domain simulation as digital signal processing functions*

a Time-marching as a state-space process
b Update equations as discrete spatial convolutions

electric conductor boundaries were modelled by a custom texture wrap mode (image processing boundary condition). Namely, out-of-bound cell coordinates, which occur in the multiresolution time-domain method when perfect electric boundaries are implemented via image theory [2], were procedurally scaled and reflected. The sequence at which the shaders are applied is illustrated in Fig. 3*b*. Note also, that tensorial values for the dielectric permittivity and conductivity of the computational domain are stored, since a generic uniaxial perfectly matched layer medium is modelled [8].

3.4 Time-to-frequency domain transformations

In several applications, such as dispersion and eigenmode analysis, the calculation of the discrete Fourier transform of the field time series, extracted by time-domain techniques, is required. To this end, a feature that enables the on-the-fly calculation of the Fourier transform of field components at a specified frequency, can readily be added to the proposed implementation of the scaling multiresolution time

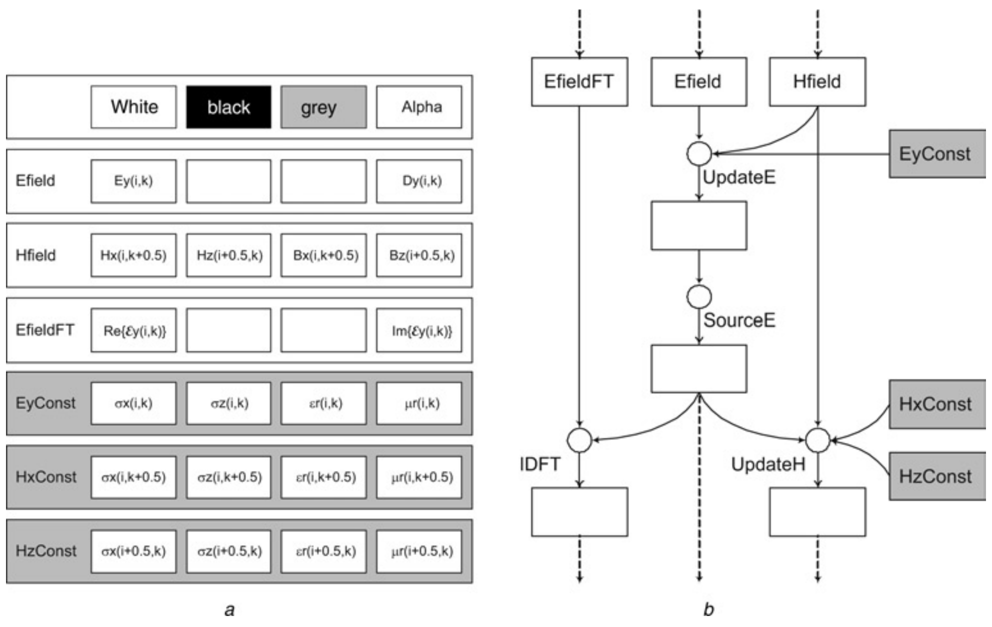


Figure 3 *Implementation of the scaling multiresolution time–domain technique for a two-dimensional uniaxial perfectly matched layer medium on a graphics processing unit*

a Texture format definitions

b Sequence of kernel passes for advancing the simulation in a single time step

domain, leveraging the performance of our graphics card accelerated simulator. Consider the Fourier transform $\mathcal{E}_y$ of the electric field component E_y at frequency f, approximated by the following expression that uses its discrete-time samples

$$\mathcal{E}_y = \mathcal{F}\{E_y\} \simeq \sum_{n=1}^{N} E_y(n\Delta t)\mathrm{e}^{\mathrm{j}2\pi f n\Delta t}\Delta t = \mathcal{F}_N \qquad (5)$$

where

$$\mathcal{F}_n = \mathcal{F}_{n-1} + E_y(n\Delta t)\mathrm{e}^{\mathrm{j}2\pi f n\Delta t}\Delta t \qquad (6)$$

Note that the iterative discrete Fourier transform of (6) is functionally similar to updating a field state in the scaling multiresolution time-domain method. Hence, it can be realised similarly to the field update equations, as shown in Fig. 3*b*. Therein, EfieldFT is a texture constituting the present cumulative sum F_n. For our particular analysis' parameterisation, it is updated once per time step and is of the same dimensions as the field textures. In relation, the iterative Fourier transform shader, denoted as IDFT in Fig. 3(b), implements the right-hand side of (6). It is responsible for evaluating $E_y(n\Delta t)\mathrm{e}^{\mathrm{j}2\pi f n\Delta t}\Delta t$ at the present time step and blending the ensuing value with the EfieldFT texture. The layout for our resulting EfieldTex texture object, including the real and imaginary part of $\mathcal{E}_y$, is presented in Fig. 3*a*.

Akin to the scaling multiresolution time domain, this realisation of the Fourier transform achieves optimal memory utilisation (introducing a scant one variable per cell) at the cost of increased computational processing per cell. By extension, it constitutes an excellent example of computation that can compound the performance advantage of graphics accelerated simulators.

4 Accuracy and performance analysis of the graphics processing unit-based scaling multiresolution time-domain technique

The accuracy and performance of our graphics accelerated simulator was evaluated against a functionally identical reference implementation. Both simulators were designed to support exactly the same inputs, outputs, data structures and operation (the number and sequence of arithmetic instructions in the implementation of update equations notwithstanding). Both were coded in C/C++ (the accelerated simulator with OpenGL and Cg, whereas the reference simulator as single threaded executable with standard libraries), and differed only in supported arithmetic precision. That is, while the accelerated simulator could only support an aberrant arithmetic for single-precision floating point (by virtue of hardware constraints), our reference simulator was coded to support all four IEEE 754 arithmetic round modes [21] (e.g. round nearest, truncate chop, round to positive infinity and round to negative infinity) for both single- and double-precision floating points. Round modes dictate how arithmetic results, so small that they fall outside the available floating-point range, are handled. Note that the standard in scientific computing is the round-nearest arithmetic round mode.

All the results presented in this section were derived on an Intel Pentium 4 2.4 GHz central processing unit with 512 MB random access memory and NVIDIA GeForce 6800 (NV40 [22]) Ultra/PCIe with 256 MB video memory. Note that, for the graphics card simulations, all memory used was on the card; there was no transfer of data to external memory involved.

4.1 Accuracy results

Under consideration was a square computational domain on the xz-plane (Fig. 4) excited by a centred transparent point source of a Gaussian pulse of the form exp $(-(t-t_o)^2/T_s^2)$, with $T_s = 1/2f_{\max}$ and $f_{\max} =$ 1 GHz (referred to, for brevity, as a 1 GHz Gaussian pulse). It was terminated in a perfect electric conductor backed uniaxial perfectly matched layer absorber and set to a fixed size of $1.28\times\ 1.28$ m^2. The mesh consisted of 512×512 cells ($\Delta x = \Delta z = 0.0025$ m) and 16 384 time steps were executed with the FDTD and Deslauriers–Dubuc schemes of orders 3, 5 and 7. The time step was chosen as

$$\Delta t = s\frac{1}{c\sqrt{1/\Delta x^2 + 1/\Delta z^2}} \tag{7}$$

where $s = 0.5$ is the well-known Courant stability factor.

A measure of accuracy was identified in the normalised Euclidean distance between the test and reference values of E_y throughout the mesh at a particular time step. This calculation is governed by (8) and involves treating each cell in an $N \times M$ mesh as an independent component of an $N \times M$-dimensional vector

$$\text{Error}_n = \frac{\sqrt{\sum_{i=0}^{N-1}\sum_{k=0}^{M-1}\left({}_nE_{i,k}^{y\text{test}} - {}_nE_{i,k}^{y\text{ref}}\right)^2}}{\sqrt{\sum_{i=0}^{N-1}\sum_{k=0}^{M-1}\left({}_nE_{i,k}^{y\text{ref}}\right)^2}} \tag{8}$$

Fig. 5 includes accuracy plots for FDTD and Deslauriers–Dubuc schemes of orders 3, 5 and 7, as a function of time steps. In each plot, the results for graphics processing unit, as well as central processing unit single-precision truncate-chop, and single-precision round-nearest simulations are depicted in white, black, and grey, respectively. Error was calculated, with respect to a central processing unit double-precision round-nearest simulation, every 16 time steps for a total of 1024 data points.

Our results show the graphics accelerated simulator exhibiting a high degree of correlation with, and consistently accruing error at a greater rate than, an equivalent single-precision round-nearest simulation. However, this error is shown to be very small, largely independent of scheme and per-time step on the order of imprecision because of an apparent arithmetic round mode. In particular, this error is closely approximated by the truncate-chop round mode.

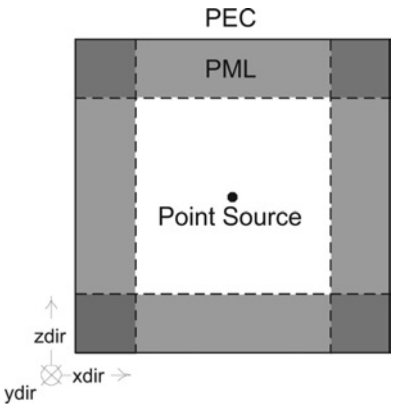


Figure 4 *Two-dimensional computational domain used for performance and accuracy tests of the graphics accelerated scaling multiresolution time–domain method*

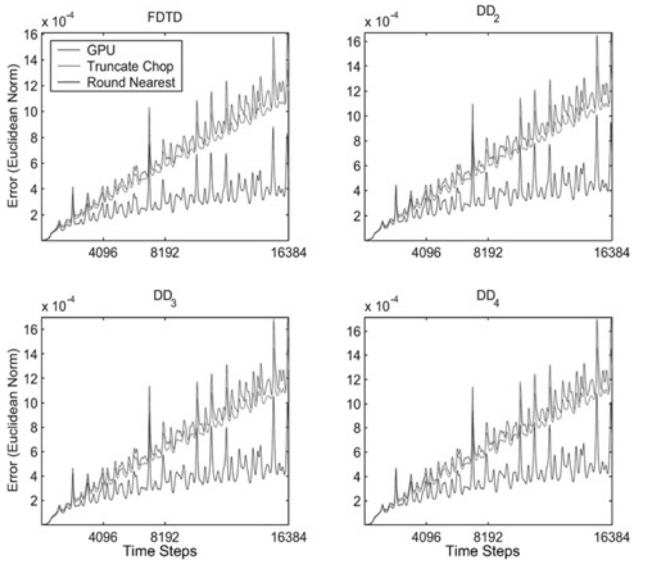


Figure 5 *Error as a function of time step for finite-difference time–domain (FDTD) and Deslauriers–Dubuc schemes of orders 2k − 1, k = 2, 3, 4 (DD$_k$) against a reference round-nearest double-precision simulation*

Table 1 Graphics processing unit (GPU), truncate-chop and round-nearest error accumulation rates (per time step, per cell) for finite-difference time–domain (FDTD) and Deslauriers–Dubuc schemes of orders $2k-1$, $k=2, 3, 4$ (DD_k)

	FDTD	DD_2	DD_3	DD_4
GPU	2.8623×10^{-13}	2.9090×10^{-13}	2.9216×10^{-13}	2.9293×10^{-13}
truncate-chop	2.5734×10^{-13}	2.6138×10^{-13}	2.6347×10^{-13}	2.6299×10^{-13}
round-nearest	9.2228×10^{-14}	1.0449×10^{-13}	1.0985×10^{-13}	1.1251×10^{-13}

Table 2 graphics processing unit (GPU) and truncate-chop relative error accumulations rates against round-nearest for finite-difference time–domain (FDTD) and Deslauriers–Dubuc schemes of orders $2k-1$, $k=2, 3, 4$ (DD_k)

	FDTD	DD_2	DD_3	DD_4
GPU/round-nearest	3.1036	2.7839	2.6596	2.6037
truncate-chop/round-nearest	2.7902	2.5014	2.3984	2.3375

Error accumulation rates are presented in Tables 1 and 2. The absolute rates (error per time step per cell) in Table 1 were determined by linear regressions for slope on the plots of Fig. 5, whereas the relative rates in Table 2 are the absolute rates in Table 1 normalised with respect to the rates of single-precision round-nearest simulations. These tables demonstrate how the inaccuracy of the graphics accelerated scaling multiresolution time-domain technique is very small and, for all intents and purposes, equivalent to an atypical round mode. In absolute terms, Table 1 shows how, as expected, the higher-order schemes universally accrue error at higher rates reflecting their increased arithmetic intensity per cell. In relation, Table 2 reveals how on average the graphics card accrues errors 2.8x faster than an equivalent single-precision round-nearest simulation; and how this rate is closely approximated by a single-precision truncate-chop simulation. These observations are entirely consistent with our card's floating-point profile introduced in [23].

Finally, and of particular note, the higher-order techniques are evidently more tolerant to floating-point imprecision than their lower-order counterparts. Table 2 shows the relative error accumulation rates decreasing with increased scheme complexity. This observation implies an interesting accuracy-based case for higher-order finite-difference schemes on graphics processing units.

4.2 Speed-up results

Under consideration was the same square domain employed in our preceding evaluation of accuracy. To evaluate the burden of mesh size on performance, here the domain was set to progressively larger dimensions of 0.16×0.16, 0.32×0.32, 0.64×0.64, 1.28×1.28, and $2.56 \times 2.56\ \mathrm{m}^2$, spatially discretised at a constant rate of $\Delta x = \Delta z = 0.0025$ m. The resulting meshes of 64×64, 128×128, 256×256, 512×512 and 1024×1024 cells progressively represent four-fold increases in memory overhead. To evaluate the relative performance of different ordered scaling multiresolution time-domain schemes, these meshes were simulated with FDTD and three Deslauriers–Dubuc schemes (of orders 3, 5 and 7) at a stability factor of $s = 0.5$. To evaluate the effect of simulation length, execution times were measured from start to 512, 1024, 2048, 4096, 8192 and 16 384 time steps.

Using the execution times of the central processing unit as baselines, we derive Fig. 6 depicting the relative speed-up achieved by the graphics processing unit. This figure shows how FDTD and scaling multiresolution time-domain schemes as a class asymptotically approach $10\times$ and $30\times$ speed-up per time step per cell. Furthermore, the graphics card

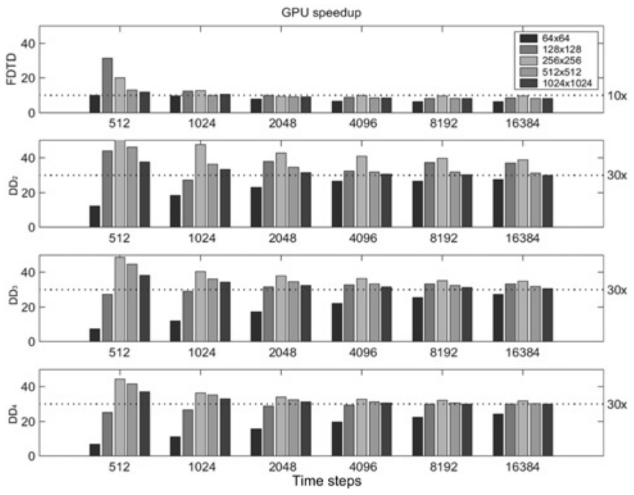


Figure 6 *Graphics processing unit speed-up for finite-difference time–domain (FDTD) and Deslauriers–Dubuc schemes of orders $2k-1$, $k=2, 3, 4$ (DD_k)*

Table 3 Spatial Resolution N_λ and resulting Δx, Δz, Mesh Size, Δt, and number of simulation steps parameterising the square computational domain simulations up to $t = 7.55$ ns with $s = 0.25$

	N_λ	$\Delta x = \Delta z$, m	Mesh size	Δt, ps	#Steps
coarse	3	0.1	64 × 64	58.93	128
medium	6	0.05	128 × 128	29.46	256
fine	12	0.025	256 × 256	14.73	512
very fine	24	0.0125	512 × 512	7.37	1024
reference	48	0.00625	1024 × 1024	3.68	2024

exhibits a comparative latency before beginning to process data. Since longer running simulations amortise this latency, it is most evident as a function of mesh size for our shortest length simulation (512 time steps). Less apparent for large meshes, the effect is on the order of the tens-to-hundreds of milliseconds and appears constant with respect to simulation scheme. With respect to Fig. 6, the latency dominates the performance of smaller meshes and accounts for their lower initial speed-ups.

4.3 Effective speed-up

Now, we simultaneously consider both performance and accuracy when the dispersion characteristics of different simulation schemes are taken into account. We confirm that the Deslauriers–Dubuc schemes enable the optimal utilisation of available memory, and (by virtue of coarser required time steps) entail fewer simulation steps to model a given problem to a set accuracy than their lower-order counterparts. Moreover, these qualities have a greater net effect on the graphics card.

Our analysis involves applying all four simulation schemes to model the same physical problem at varying spatial discretisation. Here, we considered a large $6.4 \times 6.4\ \mathrm{m}^2$ square computational domain. Again a centred 1.0 GHz Gaussian point source was used to excite the domain and the E_y field at an absolute time $t = 7.55$ ns was determined. The domain was discretised at the coarse, medium, fine, and very-fine resolutions (N_λ) listed in Table 3, and again simulated with FDTD and Deslauriers–Dubuc schemes at a constant stability factor of $s = 0.25$. The ensuing 16 trials (4 mesh sizes × 4 simulation schemes) were then studied for relative performance and accuracy.

For performance, we measured the execution times of each trial as run on the graphics and central processing unit. Observe that as a consequence of (7), coarser/smaller meshes have the secondary benefit of supporting larger time steps. That is, besides involving fewer cells to update per simulation step, fewer steps are thus required to advance a simulation a requisite amount of time. Furthermore, on the graphics processing unit, shorter simulation lengths afford a means of mitigating its systemic floating-point imprecision. The resultant mesh sizes time steps, and simulation steps parameterising trials at different spatial discretizations are summarized in Table 3.

For a measure of relative dispersion error, we compared each trial's resulting E_y field mesh with a round-nearest double-precision FDTD simulation at the reference resolution listed in Table 3. To properly compare different sized meshes, the trial meshes were up-sampled to the reference resolution with their scheme's appropriate scaling basis functions; this is possible by means of (1), which offers a continuous field approximation, once the expansion coefficients have been determined by the scaling multiresolution time-domain iterations. Then, the error was, again, calculated by the normalised Euclidean distance (8).

Table 4 Execution times (ms) observed for square computational domain simulations on the graphics processing unit, performed with the finite-difference time–domain (FDTD) and Deslauriers–Dubuc schemes of orders $2k-1$, $k = 2, 3, 4$ (DD_k)

	FDTD	DD_2	DD_3	DD_4
coarse	26.33	239.67	635.33	929.5
medium	29.33	242.5	639.5	932.33
fine	340.4	807.33	1602.33	2403.33
very fine	6217.33	11200.83	17272.33	23962.33

Table 5 As in Table 4 for simulations on the central processing unit

	FDTD	DD_2	DD_3	DD_4
coarse	182.33	920	1328.66	1750
medium	1326.33	7366	11054	14796
fine	9832	56279.33	87220.33	118535.33
very fine	73121	437155.33	683314.67	930166.33

Table 6 Dispersion error (mean normalised error per cell) evident in square computational domain simulations on the graphics processing unit, for the finite-difference time–domain (FDTD) and Deslauriers–Dubuc schemes of orders $2k - 1$, $k = 2, 3, 4$ (DD_k)

	FDTD	DD_2	DD_3	DD_4
coarse	0.846459	0.226759	0.266180	0.231539
medium	0.347350	0.069684	0.058343	0.052993
fine	0.124425	0.023564	0.022286	0.022157
very fine	0.042972	0.010024	0.009923	0.009920

Tables 4–6 comprise our results. Tables 4 and 5 list the execution times observed on the graphics and central processing unit, respectively. As expected, the former consistently outperforms the latter. In Table 4, the minimum set-up overhead of different schemes is particularly apparent in the largely constant execution times of coarse-and medium-sized meshes.

Table 6 summarises the dispersion error evident in the graphics card simulations. These values are identical, for all significant digits, to that of equivalent central processing unit simulations. For any given discretisation (across any row), error overwhelmingly improves with scheme complexity, and similarly for any given scheme (down any column), error always improves with discretisation.

The most significant result is apparent when both performance and accuracy results are considered simultaneously. Observe that in Table 6, the error evident in the medium DD4 trial is of the same order as the very-fine FDTD trial. Thus, these trials achieve largely comparable accuracy while differing in mesh size by a factor of 16 and simulation length by a factor of 4. Cross referencing these trials with the execution times in Tables 4 and 5, a net speed-up of $78.43 \times$ speed-up is evident for this particular problem to achieve the same accuracy.

5 Application: microwave indoor wireless channel modelling

This section presents the application of our graphics accelerated scaling multiresolution time-domain simulator to the modelling of an indoor ultra-wideband wireless channel. In particular, we modelled the $15.36 \times 15.36\ \text{m}^2$ floor plan illustrated in Fig. 7. Also studied in [24], it is comprised of 6–8 cm thick walls with isotropic conductivities of $\sigma = 0.002\ \Omega^{-1}$ and dielectric permittivity of $\epsilon_r = 2.89$ in an environment otherwise composed of free space. The space was discretised with $\Delta x = \Delta z = 0.02$ cm into a 768×768 cell square mesh. Sixteen layer perpendicular uniaxial perfectly matched layer absorbing boundaries were defined along all four edges and backed by perfect

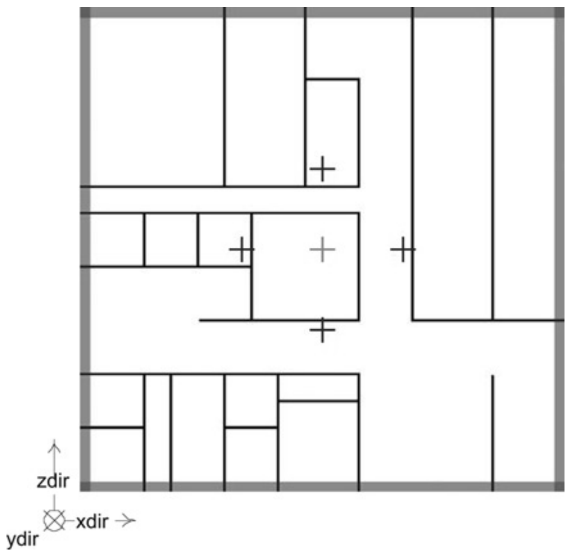
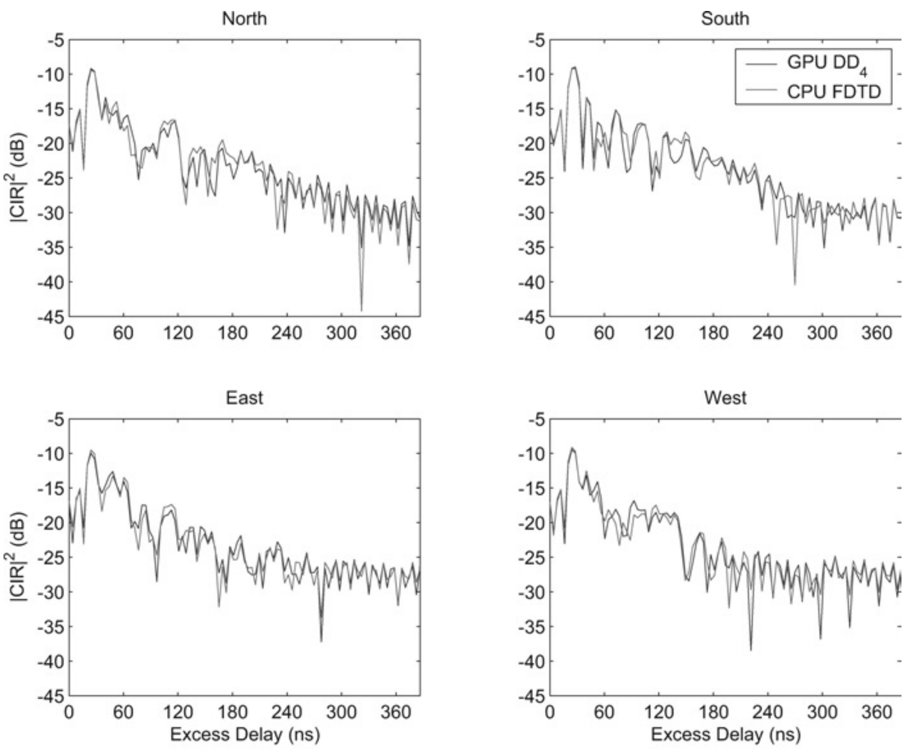
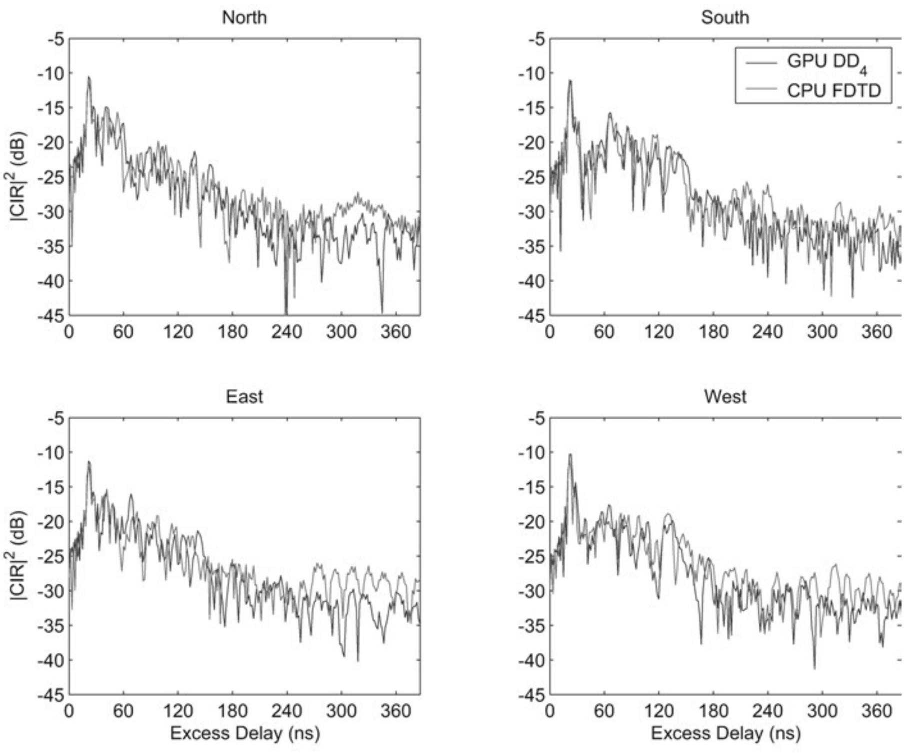


Figure 7 *A 15.36 × 15.36 m² indoor floor plan*

The source excitation and probe points, where channel impulse responses are calculated, are shown with black and grey crosses, respectively electric conductor walls. The absorber conductivity profile was chosen to have a fourth-order polynomial scaling ($m = 4$) and reflection coefficient of e^{-16}. Excitation was provided at $t = 0$ by a centred 2.4 GHz transparent Gaussian point source. Our graphics accelerated simulation employed the seventh order Deslauriers–Dubuc



a



b

Figure 8 *Apparent power-delay profiles at the four probe points in the indoor floor plan of Fig. 7*

a 1.0 GHz ultra-wideband channel
b 2.4 GHz ultra-wideband channel

Table 7 Parameters of graphics hardware accelerated Deslauriers-Dubuc scaling multiresolution time–domain (GPU DD_4) and the central processing unit finite-difference time–domain (CPU FDTD) indoor floor plan simulations

	$\Delta x/\Delta z$, m	Mesh size	UMPL cells	Δt, ps	Simulation steps	Execution time
GPU DD_4 (base)	0.02	768 × 768	16	23.59	16 384	17.25 min
CPU FDTD (reference)	0.005	3072 × 3072	64	5.897	65 536	38.53 h

Table 8 Correlation between power-delay profiles derived by the graphics hardware accelerated Deslauriers–Dubuc scaling multiresolution time-domain (base) and the central processing unit finite-difference time–domain (reference) simulations of the indoor ultra-wideband channels centred at 1.0 and 2.4 ghz

	North	South	East	West
1.0 GHz	0.98730	0.99189	0.98245	0.98660
2.4 GHz	0.91024	0.92665	0.90694	0.90831

scaling multiresolution time-domain scheme with a stability factor of $s = 0.5$. It was aimed at determining time-invariant channel impulse responses [25] at the four points illustrated in Fig. 7. Those probe points were chosen to be at a distance 2.56 m from the source (grey) at cardinal compass positions: North, South, East and West. We sought channel impulse responses for two ultra-wideband channels nominally centred at 1.0 and 2.4 GHz with bandwidths of 250 and 600 MHz (i.e. 25% of the carrier frequency), respectively.

The responses were determined post simulation in a three-step process. First, the received time-varying signals were transformed to the frequency domain. Those signals were sampled at a rate coinciding with the underlying simulator's time step Δt, over 16 364 steps, for a period of 386.44 ns. Their spectra had an ensuing 2.588 MHz resolution. Second, the spectra were normalised by the spectrum of the Gaussian source. This resulted in channel transfer functions for each channel. The ensuing transfer functions had 97 and 232 frequency–domain samples, respectively. Finally, corresponding channel impulse responses were derived by inverse Fourier transform. By extension, these had the same number of time-domain samples with excess delay bins of 4 and 1.667 ns. The channel impulse responses apparent at each of the four probe points are depicted in Figs. 8*a* and 8*b* for the 1.0 and 2.4 GHz channels, respectively. Note that these are complex valued, and their square magnitudes are termed power delay profiles [26]. Results for our simulation at the base discretisation of $\Delta x = \Delta z = 0.02$ cm are illustrated in dark grey. This discretisation amounted to spatial resolutions of $N_\lambda = 15$ and 6.25 points per wavelength at 1.0 and 2.4 GHz, respectively.

For comparison, the results of a reference FDTD simulation (executed on a central processing unit) at 4× the spatial resolution are shown in grey. Tables 7 and 8 show how with a sixteenth as many cells, and a fourth as many time steps, the graphics accelerated simulation is able to determine the power-delay profiles correlated to within 98% and 90% of the central processing unit results in 1/134 the time. It is important to note that the graphics card lacks the memory to perform such a simulation with the FDTD method, and that an equivalent scaling multiresoluton time-domain simulation on the central processing unit would only realise a 4.46× speed-up.

6 Conclusion

This paper has shown scaling multiresolution time-domain techniques to be doubly effective for graphics hardware accelerated time-domain simulations. Not only do they optimally address memory utilisation, but their classic shortcoming of increased processing per cell becomes actually advantageous on graphics hardware architectures. The dramatic sustained speed-ups at negligible inaccuracy that were demonstrated make for a very economical supercomputer, with a potentially significant impact on microwave modelling tools. In addition, the use of higher-order spatial finite differences for the discretisation of Maxwell's equations by the scaling multiresolution time domain allows for the extraction of accurate results at coarse grids, drastically reducing the overall memory requirements of large-scale problems (such as indoor wireless channel studies). This enables the graphics accelerated solution of such problems, which would be limited by memory if the FDTD were to be employed instead. The memory limitation of the FDTD method becomes even more important in three-dimensional applications, where the presented methodology can be readily extended. The study of achievable acceleration rates by the three-dimensional scaling multiresolution time-domain technique on graphics cards is a subject of ongoing work, as direct extrapolation of these rates from their two-dimensional counterparts is not possible. It is anticipated that the same factors that led to the performance advantage of this technique over the FDTD will also be even more pronounced in the three-dimensional case.

The proposed implementation was achieved by interpreting time-domain electromagnetic simulations in terms of signal processing and their update equations as discrete spatial convolution filters that were readily realised in the context of graphics processing operations. Several aspects of the implementation and the performance of the proposed technique were discussed in detail, while numerical errors involved were rigorously characterised. Finally, the simulation's very coupling with a high-performance visualisation engine inspires the possibility of real-time/interactive microwave computer-aided design and education suites [27].

7 Acknowledgment

This work has been supported by the Natural Sciences and Engineering Research Council of Canada through a Discovery Grant.

8 References

[1] TAFLOVE A., HAGNESS S.: 'Computational electrodynamics: the finite-difference time-domain method' (Artech House, Boston, MA, 2000, 2nd edn.)

[2] KRUMPHOLZ M., KATEHI L.P.: 'MRTD: new time-domain schemes based on multiresolution analysis', *IEEE Microw. Theory Tech.*, 1996, **44**, (4), pp. 555–561

[3] OWENS J.D., LUEBKE D., GOVINDARAJU N., *ET AL.*: 'A survey of general-purpose computation on graphics hardware'. Eurographics 2005, State of the Art Reports, August 2005, pp. 21–51

[4] General-purpose computing using graphics hardware website, 2006 [Online]. Available at:www.gpgpu.org

[5] HARRIS M., LUEBKE D.: 'GPGPU: general-purpose computing on graphics hardware'. SIGGRAPH Course Notes, August 2004

[6] KRAKIWSKY S.E., TURNER L.E., OKONIEWSKI M.M.: 'Graphics processor unit (GPU) acceleration of finite-difference time-domain (FDTD) algorithim'. IEEE Int. Symp. Circuits and Systems, May 2004

[7] KRAKIWSKY S.E., TURNER L.E., OKONIEWSKI M.M.: 'Acceleration of finite-difference time-domain (FDTD) using graphics processor units (GPU)'. IEEE MTT-S Int. Microwave Symp., June 2004

[8] GEDNEY S.D.: 'An anisotropic perfectly matched layer-absorbing medium for the truncation of FDTD lattices', *IEEE Trans. Antennas Propag.*, 1996, **44**, (12), pp. 1630–1639

[9] BARON G.S., SARRIS C.D., FIUME E.: 'Fast and accurate time-domain simulation with commodity graphics hardware'. Proc. Antennas and Propagation Society Int. Symp., July 2005

[10] INMAN M.J., ELSHERBENI A.Z.: 'Programming video cards for computational electromagnetics applications', *IEEE Antennas Propag. Mag.*, 2005, **47**, (6), pp. 71–78

[11] SCHNEIDER R.N., OKONIEWSKI M.M., TURNER L.E.: 'Custom hardware implementation of the finite-difference time-domain (FDTD) method'. IEEE MTT-S Int. Microwave Symp., June 2002

[12] SCHNEIDER R.N., OKONIEWSKI M.M., TURNER L.E.: 'Finite-difference time-domain method in custom hardware?', *IEEE Microw. Guided Wave Lett.*, 2002, **12**, (12), pp. 488–490

[13] DURBANO J.P., ORTIZ F.E., HUMPHREY J.R., CURT P.F., PRATHER D.W.: 'FPGA-based acceleration of the 3D finite-difference time-domain method'. FCCM, 2004, pp. 156–163

[14] DURBANO J.P., ORTIZ F.E., HUMPHREY J.R., CURT P.F., PRATHER D.W.: 'Hardware implementation of a three-dimensional finite-difference time-domain algorithm', *IEEE Antennas Wirel. Propag. Lett.*, 2003, **2**, (1), pp. 54–57

[15] VERDUCCI L., PLACIDI P., CIAMPOLINI P., SCORZONI A., ROSELLI L.: 'A standard cell hardware implementation for finite-difference time domain (FDTD) calculation'. Proc. IEEE MTT-S Int. Microwave Symp. Digest, 2003, vol. 3

[16] BARON G.S., FIUME E., SARRIS C.D.: 'Accelerated implementation of the S-MRTD technique using graphics processor units'. IEEE MTT-S Int. Microwave Symp., 2006

[17] FUJII M., HOEFER W.J.R.: 'Application of biorthogonal interpolating wavelets to the Galerkin scheme of time dependent Maxwell's equations', *IEEE Microw. Wirel. Compon. Lett.*, 2001, **11**, (1), pp. 22–24

[18] FUJII M., HOEFER W.J.R.: 'Dispersion of time domain wavelet Galerkin method based on Daubechies' compactly supported scaling functions with three and four vanishing moments', *IEEE Microw. Guided Wave Lett.*, 2000, **4**, (0), pp. 125–127

[19] KOVVALI N., LIN W., CARIN L.: 'Order of accuracy analysis for multiresolution time-domain using daubechies bases', *Microw. Opt. Tech. Lett.*, 2005, **45**, (4), pp. 290–293

[20] SGI: 'OpenGL Extension Registry', 2006, see: NV_fragment_program2, NV_float_buffer, WGL_ATI_pixel_format_float, and WGL_ARB_pbuffer

[21] ANSI/IEEE : 'IEEE Standard for Binary Floating-Point Arithmetic'. ANSI/IEEE, August 1985, std pp. 754–1985

[22] NVIDIA: "GeForce 6800,' Product Overiew, 2004 [Online]. Available at: www.nvidia.com/page/geforce_6800.html

[23] HILLESLAND K., LASTRA A.: 'GPU floating point paranoia'. GP2: ACM Workshop on General Purpose Computing on Graphics Processors, August 2004

[24] ALIGHANBARI A., SARRIS C.D.: 'Simulation of wireless channels via biorthogonal interpolating function-based high order S-MRTD time domain techniques'. Annual Review of Progress in Applied Computational Electrodynamics, March 2006

[25] HASHEMI H.: 'The indoor radio propagation channel', *Proc. IEEE*, 1993, **81**, (7), pp. 943–968

[26] RAPPAPORT T.S.: 'Wireless communications: principles and practice' (Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001)

[27] LEUNG M., LEUNG J., BARON G.S., SARRIS C.D.: 'A fast time-domain wireless channel simulation tool for radio-wave propagation courses'. Proc. IEEE Int. Symp. Antennas Propagation, 2006