Progress In Electromagnetics Research M, Vol. 20, 115–126, 2011

FAST COMPUTATION OF DIPOLE RADIATION IN STRATIFIED BACKGROUND USING GRAPHICS PROCESSING UNIT

M. Quinto, S. Boutami *, and J. Hazart

CEA-Leti, MINATEC Campus, 17 rue des Martyrs, 38054 Grenoble Cedex 9, France

Abstract—We present the GPUs computation acceleration for a very recurrent electromagnetic problem which is the calculation of the field radiated by electric dipoles in a multilayer structure (Green's tensor in stratified background), based on the well-known Sommerfeld integrals. Using an optimized parallelization scheme, huge computation acceleration is obtained. Applications of such a work are very broad, especially for the modeling of stratified light emitting devices, or as a building block for the calculation of optical scattering by complex shape structures, when using methods as discrete dipole approximation (DDA) or method of moments (MoM) for example.

1. INTRODUCTION

Simulations play an important role in research environment, in order to predict or explain physical phenomena. However, simulations may sometimes require very large memory and time in computation. This is particularly the case with full-wave three-dimensional optical simulations, which consist in solving Maxwell equation in 3D space. The multilayer Green's tensor is, in this context, a powerful tool, for two main reasons: first, it allows by itself to express the field radiated by elementary emitters in a planar multilayer background, which is directly of great interest for the simulation of emitting devices, such as light emitting diodes (LEDs) for example. Secondly, it allows the simulation of complex-shape nanostructures in a multilayer background. Indeed, many electromagnetic methods consist in decomposing a complex-shape object scattering by several elementary dipoles. This is the case, for example, with the discrete dipole

Received 25 May 2011, Accepted 23 July 2011, Scheduled 19 August 2011

* Corresponding author: Salim Boutami (salim.boutami@cea.fr).

approximation (DDA) [1], or the method of moments (MoM) [2, 3]. These methods require to calculate the radiation of a single dipole on several spatial points. When the background medium is homogeneous, the dipole radiation is straightforward, the Green's tensor being known analytically, by the means of spherical Hankel function [3]. However, in most applications, a stratified environment is present (at least, the device is fabricated on a substrate). This is due to the fact that optical devices have benefited from the planar technology available from microelectronics. The stratified Green's tensor could be calculated using numerical methods as finite-difference time domain method [4]. However, if the number of layers is huge, or if some layers are thin or metallic, the number of mesh cells can be very high if a good accuracy is desired, leading to high memory requirements in 3D. Moreover, perfect matching layers [5] are needed to avoid reflection of radiative as well as guided modes at the boundary of the necessarily truncated calculation domain. A nice way to circumvent these issues is to use a semi-analytical formulation for the calculation of a dipole radiation in a stratified environment, known as Sommerfeld integral [6–19]. This integral consists in decomposing the dipole radiation into plane waves, since the plane wave is the canonical case for which propagation is analytically known in stratified media, by the use of Fresnel coefficients. In a general case, Sommerfeld integrals have no closed-form solution and must therefore be computed numerically. The advantage of this approach is that no mesh of the layers is needed, and the plane wave formalism is able to handle laterally infinite layers.

In order to know a field component radiated by a dipole in space, one has to perform the numerical evaluation of a Sommerfeld integral for each point of space where this component is wanted. So, when one has many dipoles and many spatial points for field calculation, which is generally the case with DDA or MOM, many Sommerfeld integrals have to be computed (typically several tenths of thousands for a scattering object of the wavelength-scale), which is very time-consuming on a central processing unit (CPU) (at least several tenths of minutes on a standard CPU). As Sommerfeld integrals for different spatial points can be calculated independently, the Green's tensor problem is highly parallelizable, and it may be advantageous to use the massive parallel calculation of GPUs for this problem.

Previous experimentations of GPUs for computational electromagnetics concerned essentially physical optics, geometric optics [20, 21], or FDTD [22–25]. A few works have been devoted more recently to the acceleration of MoM by GPU [26–28]. The present work shows, for the first time to our knowledge, that GPUs can be highly efficient for Sommerfeld integral calculation. It represents therefore a

new step in the emerging area of GPU-computational EM and more particularly GPU-enhanced MOM.

2. OPTICAL MODEL

The radiation of an arbitrary oriented dipole in a multilayer can be obtained by the projection of its orientation on the three main orthogonal directions of space, using the Green's tensor, which represents the electric field radiated by three orthogonal dipoles. As a flat multilayer is rotationally invariant with respect to any axis perpendicular to interfaces (which will be the z direction in the following), the Green's tensor can be resumed to the calculation of the field radiated by a vertical dipole and an arbitrary horizontal dipole. The whole formalism for Green's tensor in stratified media can be found, for example, in [13]. Our paper will focus on GPUs for such a formalism. In this paper, we will show the example of a vertical dipole; although the formulation for a horizontal dipole is slightly different, it leads to Sommerfeld integrals which can be treated in the same way.

A vertical dipole radiates 6 field components: E_x, E_y, E_z (electric field components) and H_x, H_y, H_z (magnetic field components) in space. The field components involve, in their corresponding Sommerfeld integrals, Bessel function of different orders. For example, the Sommerfeld expression of the E_z component, scattered by the multilayer interfaces in a spatial point of the layer m and due to a vertical dipole located in medium l, can be written [13]:

$$E_z(\rho, z) = \int_0^\infty \frac{k_\rho^3}{k_m^2 k_{mz}} J_0(k_\rho \rho) \left[A(k_\rho) \cdot e^{jk_{mz}z} + B(k_\rho) \cdot e^{-jk_{mz}z} \right] dk_\rho \quad (1)$$

where ρ is the horizontal projection of the distance between the spatial point of observation and the dipole (see Figure 1); k_m is the wavevector in medium m; k_{mz} is the vertical projection of the wavevector in medium m. More precisely, k_{mz} is defined as the following square root, chosen with positive imaginary part in substrate and superstrate, ensuring a decay of fields at infinity:

$$k_{mz} = \sqrt{k_m^2 - k_\rho^2} \quad (2)$$

J_0 is the Bessel's function of the first type and order 0. The terms $A(k_\rho)$ and $B(k_\rho)$ are the generalized multilayer reflection and transmission coefficients, based on Fresnel's coefficient, which satisfy field continuity conditions at all interfaces of the multilayer, and therefore depend on k_ρ wavevector, all layer thicknesses and refractive indexes, the position of the dipole in the multilayer, but also the kind of polarization of the

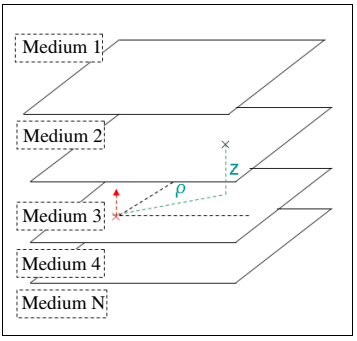

Figure 1. Schematic of a multilayer environment, with the cylindrical coordinates implied in the sommerfeld integral (1); the dipole is in medium $l = 3$, whereas the spatial point where field is calculated is in layer $m = 2$.

plane wave considered. These generalized coefficients can be iteratively obtained in all layers by a transfert matrix method, also described in [13]. One should note that, for a vertical dipole, only p-polarisation should be considered [14]. The variable k_ρ, which is the variable of integration, is the horizontal projection of the wavevector of an elementary plane wave composing the dipole radiation. The structure is rotationally symmetric with respect to the z-axis, which is why it is possible to perform a mono-dimensional integration over k_ρ, avoiding a double integration on both k_x and k_y planar wavevector components.

The Sommefeld integral numerical evaluation is not trivial, due to some possible singularities (example when k_ρ matches a guided mode of the multilayer, related to a pole of the integrand), and branch and point cuts (when $\mathrm{Im}\,(k_{mz}) = 0$, condition related to radiated modes in substrate and superstrate). Thus, it is not possible to solve it on the real axis. But thanks to the Cauchy's theorem, we can deform the integration path in the complex plane, avoiding singularities, in a way similar to [13] (see Figure 2). The optimal numerical method to perform the Sommerfeld integral is known as Gauss-Kronrod method, which can handle the highly oscillatory behavior of the integrand [13].

3. GPU AND CUDA

The new design of GPUs allows them to be very suitable not only for graphic tasks (video-games, etc.) but also for scientific task as hardware accelerator [29].

In the last years, GPU performances have been increasing much more than CPU performances. This gap relies on the difference in

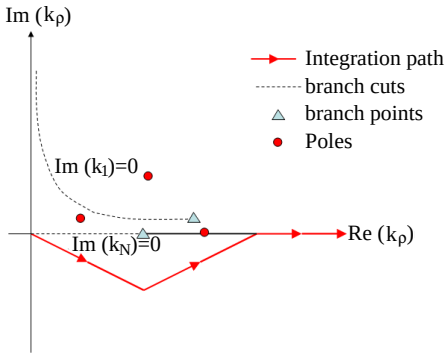

Figure 2. Deformation of the integration path for sommerfeld integral calculations, in order to avoid poles, and branch cuts and points. Branch cuts and points are only related to the susbtrate (medium 1, supposed absorbing in this example) and superstrate (medium N, supposed non-absorbing) of the multilayer.

architecture between them. In fact, the CPU architecture is optimized for sequential code and has a sophisticated control logic and large cache memories which reduce the instruction and data access latency of the different tasks, while the GPU architecture is designed to perform the same execution on a huge amount of data. Moreover, the success of the GPU in scientific environment is particularly due to Computed Unified Device Architecture (CUDA) [30]. CUDA is a tool developed by Nvidia and based on C language, for parallel computing architecture, which allows users to manage a GPU as a coprocessor. The GPU is represented as a set of multiprocessor as Single Instruction Multiple Data (SIMD). Each SIMD executes the same instruction on different data. The main feature of CUDA is that the programmer does not care about threads scheduling. However, the programmer must be careful of the shared memory management. The shared memory available on GPU is limited to 16 KB, for the Nvidia G80 series. The optimization of the share memory has consequences on the speed-up factor.

The hardware used for this study is:

- Core (TM) 2 Duo CPU E6750 @2.66 GHz 64-bit, 4 GB of RAM, operating system Red Hat Enterprise 5.3 64-bit;
- Nvidia GeForce series 8800 GTX v1.0.

4. PARALLEL ALGORITHM

Before explaining the whole parallel algorithm, we would like to give some clarifications to the reader about the Gauss-Kronrod algorithm

for numerical evaluation of a Sommerfeld integral. We translated and adapted the Matlab *quadgk* function [31], which is based on the Gauss-Kronrod (7,15) numerical integration method. The *quadgk* Matlab function performs a generic integral in a recursive way (adaptive algorithm). It divides the whole integration path in a fix number of subinterval, and for each of them, it applies the Gauss-Kronrod rule. In the general case, each subinterval is sampled by 150 points in order to evaluate the integral on this subinterval. But, if 150 points are not enough to obtain the desired accuracy, the subinterval is divided in a recursive way in more subintervals, and so on. For the Bessel's function computation, we implemented, for both CPU and GPU, the routines for special function made by Zhang and Jin [32].

Concerning now the parallelization scheme used for the Sommerfeld integrals calculations, in order to evaluate the field reflected and transmitted by the dipole in a certain region of space discretized in several spatial points, we need to perform two loops: one on the integration variable k_ρ (which corresponds to the different plane waves composing the dipole radiation) and the other on the space points where we want to calculate the field. The parallelized threads (executing the same operation on different data) on GPU can be therefore defined in several manners:

1. The threads calculate a Sommerfeld integral on different spatial points.
2. Sequential CPU loop on the integration variable k_ρ, the threads calculates the influence of one single k_ρ-plane wave on different spatial points.
3. Sequential CPU loop on the spatial points, each thread calculates the influence of the different plane waves on one single spatial point.

In the first solution, the Sommerfeld integrals are parallelized, but this requires too much memory (register and shared memory on GPU) to execute (we need to store for each thread the contribution of 150 plane waves). The last solution consists in doing a sequential CPU loop on the spatial points and making a GPU parallelization on the k_ρ values, but the latter are very few (150). Consequently, the potentiality of GPU is unused. These observations led us to adopt the solution 2, which appears non-intuitive, but is the most efficient. A CPU loop is created on the k_ρ values, and for each of them the integrand is partially evaluated on all spatial points (several thousands of points). Thus, all the massive parallel-computing effort of GPU is employed. The results are stored in a cumulative vector incremented at each iteration, in order to progressively reconstruct the Sommerfeld integrals.

To summarize, the execution mode of the final algorithm is shown in the flowchart of Figure 3.

As shown by the flowchart, there are five different functions inside the k_ρ loop, executed on GPU, called kernel 1-5, which perform the following tasks:

- kernel 1: Bessel's function computing (use ρ-coordinate of spatial points).
- kernel 2: exponential function computing (use z-coordinate of spatial points).
- kernel 3: multiplies the two previous results, with the generalized reflection and transmission multilayer coefficient previously

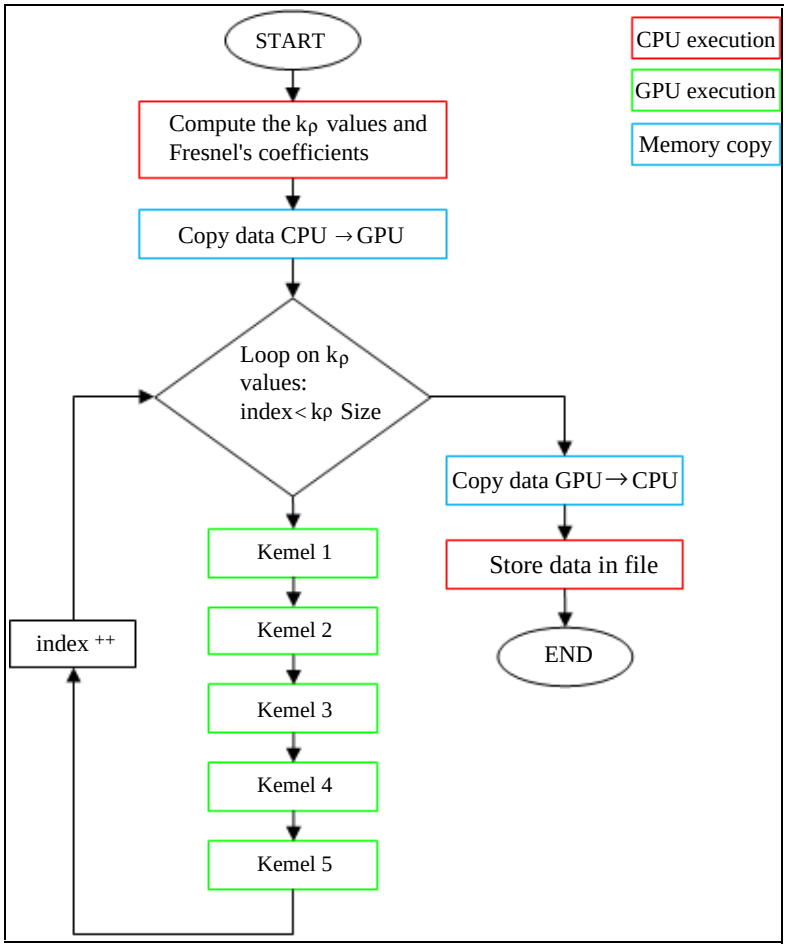

Figure 3. Flowchart of the parallel algorithm.

calculated on CPU, and the weights related to k_ρ in the Gauss-Kronrod numerical integration.

- kernel 4: store the values in a cumulative vector.
- kernel 5: initialize to 0 the vectors used by kernel 1 and 2 in order that they should be used in the next iteration.

5. RESULTS

In this section we evaluate the gain in computation time obtained by the use of GPU. We performed, as an example, the calculation of the field radiated by a vertical electric dipole (see Figure 4) in a two-media space. The dipole is located in glass (refractive index $n = 1.5$), at 10 nm from the interface with air and radiates at wavelength $\lambda = 600$ nm.

Table 1 shows summarized computation times for both CPU (Matlab) and GPU (stand alone) algorithms, as a function of the number of evaluation points. Besides, a MEX-file [33, 34] was created to call the GPU algorithm from Matlab. Thanks to MEX-file, the whole algorithm written in C programming language can be called from Matlab environment like a Matlab function, with the advantage that it is still executed on GPU. As can be seen in Table 1, when the GPU is called from Matlab, the execution time is slightly different, due to overhead introduced by Matlab.

Figure 5 shows the electromagnetic field radiated by the vertical dipole. The maximum value of absolute error, as shown in Figure 6, is of the order of magnitude the *quadgk* function precision, when working

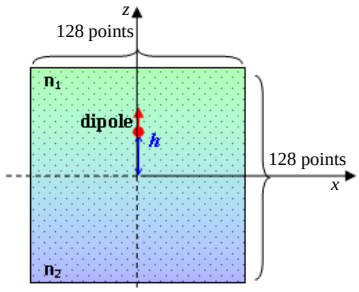

Figure 4. Numerical test: calculation of the norm of the reflected and transmitted Ez field, due to the radiation of a vertical dipole located in glass (top medium), at a distance $h = 10$ nm from the interface with air (bottom medium), for a wavelength $\lambda = 600$ nm. The resolution of spatial points where field is calculated is 128×128, spaced by 1 nm, and centered in (0,0) in the XZ plane ($Y = 0$). The spatial resolution is then increased, the spatial range being kept constant.

Table 1. Result of the computation time.

Number of spatial points	Computation Time			Corresponding Gains	
	Matlab [s]	GPU stand alone [s]	GPU + Matlab [s]	Matlab/(GPU stand alone)	Matlab/(GPU+Matlab)
16384	83	0.183	0.255	~450	~370
65536	342	0.626	0.685	~550	~500
262144	1382	2.369	2.654	~580	~520
1048576	5718	9.483	9.853	~600	~580

All the computation times are averaged over 10 executions. The incertitude error on the GPU execution time is around ± 0.3 ms.

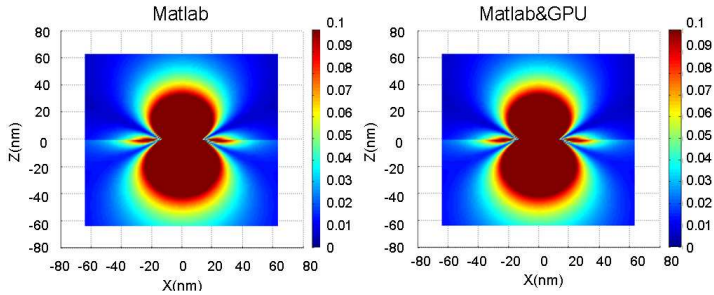

Figure 5. Results for the configuration of figure 4, for Matlab (a), and Matlab with GPU (b).

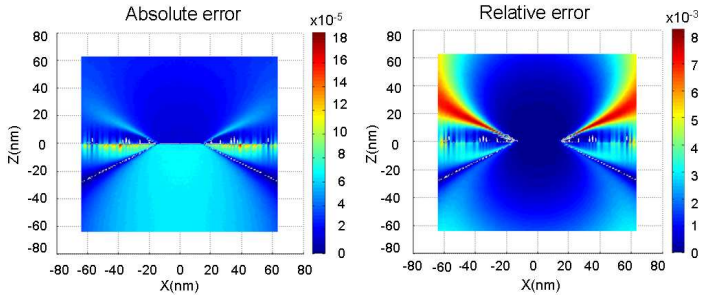

Figure 6. Absolute and Relative errors between Matlab, and Matlab with GPU.

with floating-point data, i.e., 10^{-4} [31]. Better accuracy could be obtained when moving towards double precision GPUs.

6. CONCLUSIONS

The gain obtained in computation time is very important (from several hours for CPU calculations, to a few seconds for GPU). We show that

GPU is an excellent solution compared to cluster or supercomputer, for highly parallelizable optical algorithms such as Sommerfeld integrals calculation, when advantageous parallelization strategy is adopted. This work paves the way to very fast electromagnetic simulations of multilayer light emitting devices or very fast calculation of scattering by complex shape objects embedded in multilayers, in semi-analytical electromagnetic methods using multilayer Green's function, as DDA or MOM.

REFERENCES

1. Purcell, E. M. and C. R. Pennypacker, "Scattering and absorption of light by nonspherical dielectric grains," *Astrophysical Journal*, Vol. 186, 705, 1973
2. Harrington, R. F., *Field Computation by Moment Methods*, Krieger Publishing Co., Inc., 1968.
3. Hafner, C., *The Generalized Multipole Technique for Computational Electromagnetics*, Artech, 1990.
4. Yee, K., "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, Vol. 14, 302–307, 1966.
5. Berenger, J. P., "A perfectly matched layer for the absorption of electromagnetic waves," *J. Comput. Phys.*, Vol. 114, 185–200, 1994.
6. Sommerfeld, A., "Propagation of waves in wireless telegraphy," *Ann. Phys. (Leipzig)*, Vol. 28, 665–737, 1909.
7. Michalski, K. A. and J. R. Mosig, "Multilayered media Green's functions in integral equation formulations," *IEEE Transactions on Antennas and Propagation*, Vol. 45, No. 3, 508–519, March 1997.
8. Barkeshli, S., P. H. Pathak, and M. Marin, "An asymptotic closed-form microstrip surface Green's function for the efficient moment method analysis of mutual coupling in microstrip antenna," *IEEE Transactions on Antennas and Propagation*, Vol. 38, No. 9, 1374–1383, September 1990.
9. Ayatollahi, M. and S. Safavi-Naeini, "A new representation for the Green's function of multilayer media based on plane wave expansion," *IEEE Transactions on Antennas and Propagation*, Vol. 52, No. 6, 1548–1557, June 2004.
10. Chow, Y. L., J. J. Yang, D. G. Fang, and G. E. Howard, "A closed-form spatial Green's function for the thick microstrip substrate,"

IEEE Trans Microw. Theory Tech., Vol. 39, No. 3, 588–592, March 1991.

11. Ling, F. and J.-M. Jin, "Discrete complex image method for Green's functions of general multilayer media," *IEEE Microw. Guided wave Lett.*, Vol. 10, 400–402, October 2000.
12. Yuan, M. and T. K. Sarkar, "Computation of the Sommerfeld integral tails using the matrix pencil method," *IEEE Transactions on Antennas and Propagation*, Vol. 54, No. 4, 1358, April 2006.
13. Paulus, M., P. Gay-Balmaz, and O. J. F. Martin, "Accurate and efficient computation of the Green's tensor for stratified media," *Physical Review E*, Vol. 62, No. 4, 5797–5807, October 2000.
14. Novotny, L., "Allowed and forbidden light in near-field optics in a single dipolar light source," *JOSA B*, Vol. 14, 91–104, 1997.
15. Xu, X.-B. and Y. F. Huang, "An efficient analysis of vertical dipole antennas above a lossy half-space," *Progress In Electromagnetics Research*, Vol. 74, 353–377, 2007.
16. Jarchi, S., J. Rashed-Mohassel, and R. Faraji-Dana, "Analysis of microstrip dipole antennas on a layered metamaterial substrate," *Journal of Electromagnetic Waves and Applications*, Vol. 24, No. 5–6, 755–764, 2010.
17. Parise, M., "Exact electromagnetic field excited by a vertical magnetic dipole on the surface of a lossy half-space," *Progress In Electromagnetics Research B*, Vol. 23, 69–82, 2010.
18. Poljak, D. and V. Doric, "Wire antenna of simple grounding systems, Part I: The vertical grounding electrode," *Progress In Electromagnetics Research*, Vol. 64, 149–166, 2006.
19. Poljak, D. and V. Doric, "Wire antenna model for transient analysis of simple grounding systems, Part II: The horizontal grounding electrode," *Progress In Electromagnetics Research*, Vol. 64, 167–189, 2006.
20. Gao, P. C., Y.-B. Tao, and H. Lin, "Fast RCS prediction using multiresolution shooting and bouncing ray method on the GPU," *Progress In Electromagnetics Research*, Vol. 107, 187–202, 2010.
21. Tao, Y.-B., H. Lin, and H. J. Bao, "From CPU to GPU: GPU-based electromagnetic computing (GPUECO)," *Progress In Electromagnetics Research*, Vol. 81, 1–19, 2008.
22. Zainud-Deen, S. H., E. Hassan, M. S. Ibrahim, K. H. Awadalla, and A. Z. Botros, "Electromagnetic scattering using GPU-based finite difference frequency domain method," *Progress In Electromagnetics Research B*, Vol. 16, 351–369, 2009.
23. Xu, K., Z. Fan, D.-Z. Ding, and R.-S. Chen, "GPU accelerated

unconditionally stable Crank-Nicolson FDTD method for the analysis of three-dimensional microwave circuits," *Progress In Electromagnetics Research*, Vol. 102, 381–395, 2010.

24. Tay, W. C., D. Y. Heh, and E. L. Tan, "GPU-accelerated fundamental ADI-FDTD with complex frequency shifted convolutional perfectly matched layer," *Progress In Electromagnetics Research M*, Vol. 14, 177–192, 2010.
25. De Donno, D., A. Esposito, and L. Tarricone, "Introduction to GPU computing and CUDA programming: A case study on FDTD," *IEEE Antennas and Propagation Magazine*, Vol. 52, No. 3, 116–122, June 2010.
26. De Donno, D., A. Esposito, G. Monti, and L. Tarricone, "Parallel efficient method of moments exploiting graphics processing units," *Microwave and Optical Technology Letters*, Vol. 52, No. 11, 2568–2572, November 2010.
27. De Donno, D., A. Esposito, G. Monti, and L. Tarricone, "GPU-based acceleration of MPIE/MoM matrix calculation for the analysis of microstrip circuits," *The 5th European Conference on Antennas and Propagation (EuCAP)*, 1–15, Rome, Italy, April 2011.
28. Peng, S. and N. Zaiping, "Acceleration of the method of moments calculations by using graphics processing units," *IEEE Transactions on Antennas and Propagation*, Vol. 56, No. 7, 2130–2133, July 2008.
29. Halfhill, T. R., "Parallel processing with CUDA," *Microprocessor Report*, 2008.
30. Nvidia, "Programming guide version 1.0," 2007, http://developer.download.nvidia.com/compute/cuda/1_0/NVIDIA_CUDA_Programming_Guide_1.0.pdf.
31. MatlabWorks, "Numerically evaluate integral, adaptive Gauss-Kronrod quadrature," http://www.mathworks.com/access/helpdesk/help/techdoc/ref/ quadgk.html.
32. Zhang, S. and J. Jin, "Computation of special functions," http://jin.ece.uiuc.edu/routines/routines.html.
33. MathWorks, "Mex-file guide," http://www.mathworks.com/support/tech-notes/1600/1605.html.
34. Nvidia, "Accelerating matlab with cuda using mex file," http://www.developer.download.nvidia.com/compute/cuda/1_0/AcceleratingtextdiscountMatlab\discount20with\textdiscount20-CUDA.pdf.