Benefits and Challenges of GPU Accelerated Electromagnetic Solvers from a Commercial Point of View

Ulrich Jakobus

Altair Development S.A. (Pty) Ltd.
Stellenbosch, 7600, South Africa
jakobus@altair.com

Abstract ─ This paper discusses the benefits but also challenges of GPU accelerated electromagnetic solvers from a commercial point of view, namely using FEKO as example. Specifically, the effects of some of the complex interdependencies between different components are presented. It is shown that despite the advances made in the field of GPGPU computing, and impressive speedups for parts of a program or simplified problems, there are a number of factors to consider before these techniques can be applied to a commercial product that is expected to be robust and, most importantly, to always give trustworthy results for a wide variety of problems.

Index Terms ─ Commercial Solvers, CUDA, FDTD, FEKO, FEM, GPGPU, GPU Acceleration, MoM, RL-GO, SBR.

I. INTRODUCTION

In the field of computational electromagnetics (CEM), a wide range of numerical techniques can be used to simulate a variety electromagnetic radiation and scattering problems. One of the primary reasons that such a wide variety of methods exists, is that no single method performs best for all problem types [1]. Thus, one of the first challenges in solving an electromagnetic problem is to select the method that is best or at least reasonably suited to the problem of interest.

Even with the optimal method selected, there is still the matter of the available computational resources to consider. It may then be that the desired solution takes hours, days, or even weeks to compute. One of the ways in which an attempt has been made to increase the computational power at disposal – thereby decreasing the time required for a solution – has been to make use of graphics processing units (GPUs) to perform general purpose computational tasks, and not just the graphics-related tasks for which they were originally designed for. This practice, called general purpose GPU (GPGPU) computing, has seen a remarkable increase of late, both in terms of hardware capability, as well as the ease with which these devices can be programmed [2].

The most common way of programming such devices is using the Compute Unified Device Architecture (CUDA) by NVIDIA. This couples a genuinely programmable hardware architecture with programming tools that can be used by any developer with a knowledge of C/C++. Previously, GPGPU programming involved convincing a GPU to do what one wanted by rewriting computational routines as graphics programs. Since its inception, CUDA's hardware/software combination has evolved to such an extent that the latest generation of devices can be found in the fastest supercomputers in the world, with a much more powerful set of software features available as well.

There has been considerable development and a large number of papers were published on the GPU acceleration of CEM methods, for example the Method of Moments (MoM) [3] and [4], the Finite Element Method (FEM) [5] and [6], and the method of Shooting and Bouncing Rays (SBR) [7] and [8]. More general advances such as in GPU based dense linear algebra methods can be found, e.g., in [9]. The focus of this paper is not to add to this (we have done so earlier, e.g., in [10] or [11]), but instead to present an alternate perspective on these advances. That is to say the use of GPU technology as well as the challenges related to it are considered from the point of view of a commercial CEM software. To this end, the software package FEKO [12] is taken as an example. The motivation for this is that quite often such advances are considered from a purely academic standpoint, and this leads to a number of short-comings and challenges being overlooked.

Section II gives a short introduction on the FEKO solution kernel and the various CEM methods that are supported by it. This serves as background for a discussion on the difficulties associated with the GPU acceleration of a commercial CEM software package such as FEKO in Section III, and a short discussion of GPU accelerated solvers that exist in FEKO or are under development in Section IV. The paper is concluded in Section V, where a discussion on future paths to facilitate further GPU acceleration is included.

Submitted On: August 28, 2015
Accepted On: January 24, 2016

II. THE FEKO SOLUTION KERNEL

As already mentioned, a number of CEM methods exist which have their own strengths and weaknesses, and which can solve various problems of interest with varying degrees of success. It is thus important that a commercial CEM code such as FEKO implements a number of these methods to allow it to be competitive for a large selection of target application areas.

Figure 1 shows the various solution techniques available in FEKO for the solution of RF/microwave problems. Two factors influencing the choice of solution method – the electrical size of the problem being considered and the complexity of the materials being simulated – are indicated on the axes. The possibility of hybridizing various methods exists, and this allows for the solution of more complex problems by selecting the best solution method for different regions of the same problem with full bi-directional coupling between them.

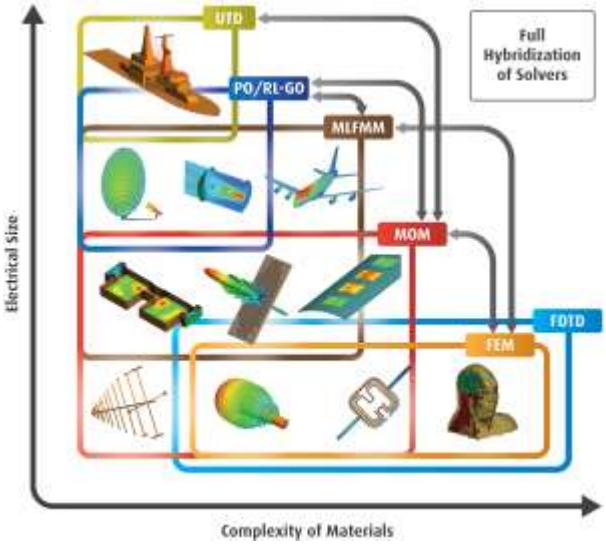

Fig. 1. A diagram depicting the various computational methods in FEKO. The hybridization that exists between some of the methods is also shown by green arrows.

III. CHALLENGES IN GPU ACCELERATION

In any software development, it is required that the available resources be allocated to maximize the delivered value in the software project. How value is determined is specific to each project, and may also differ greatly between the academic and commercial environments. In the commercial environment, for example, the number of customers with capable hardware demanding or being able to use GPU acceleration directly influences the relative value of GPU accelerated extensions when compared to other feature extensions. Academic development may, on the other hand, place a high importance on novelty for use in academic publications.

A. Versatility, reliability, and reproducibility

Many academic publications on the topic of accelerated CEM codes consider a small number of examples to illustrate the applicability or performance improvements of a specific method. These examples are often simple or canonical problems, which may play to the strengths of the method being considered, and also may not exceed the resources – such as available memory – of the GPU being used for acceleration.

In the commercial setting, there is no such control over which examples are being considered, and customers expect accurate results for a wide variety of problems. This not only imposes heavy resource requirements for additional validation and verification of the accelerated methods, but also in the detection of possible problem cases at run-time (such as running out of GPU memory and then switching automatically to block based algorithms or switching the computations on the fly back to the CPU), and handling these in a well-rounded and user friendly way.

B. Variety of CEM methods

The various computational methods included in the FEKO solution kernel and discussed in Section II have their own strengths and weaknesses when it comes to the solution of CEM problems. In addition, each of these methods present its own challenges in parallelization in general (MPI, OpenMP, etc.), and in GPU computing specifically.

Take the Methods of Moment (MoM) and the Finite Element Method (FEM) as examples. These are both matrix-based methods which require the construction, and (for driven problems) the subsequent solution of a linear system of equations. It is also possible to formulate certain classes of problems in each method as generalized eigenvalue problems.

At this point it may seem as if these two methods would be amenable to similar approaches when considering them for GPU acceleration. The situation is, however, that the linear system which results as part of a MoM computation is dense, whereas that associated with the FEM is a sparse system. Although GPU tools exit for the solution of both types of systems, the difference in performance of dense and sparse computation on a GPU means that the realized speedup will differ significantly. Furthermore, the effect of the other phases in the solution process (e.g., matrix fill) must also be taken into consideration and will be discussed in Section IV.

C. Software and design decisions

Another important factor regarding the adoption of GPU acceleration in an existing commercial CEM package are design and development decisions such as the language of implementation and low-level program

flow, which if – if not selected carefully – may not map well to massively parallel architectures such as GPUs.

CUDA was already mentioned as programming language to support NVIDIA cards. In the OpenMP 4.0 standard, for example, provision has been made for the use of accelerators. OpenMP is a directive-based, open standard which provides a portable means to parallelize code over a number of threads. The inclusion of the concept of accelerators and the associated operations, means that the importance of such technologies has been recognized. Furthermore, since the directives are platform agnostic, acceleration would in theory not be limited to a particular set of devices – such as NVIDIA GPUs when using CUDA – but the same code could be used to run on multi-core CPUs, GPUs by other vendors, and other accelerator technologies such as Intel's Xeon Phi coprocessors. There is also OpenCL, kind of being in the middle between CUDA and OpenMP. In FEKO, all three techniques (OpenCL, OpenMP, and CUDA) are being explored and partially used, but all the following GPU discussions refer to CUDA specifically.

Considering that many of the GPGPU programming tools are centered on C/C++ implementations, the options for the acceleration of for instance FORTRAN based routines generally involve rewriting large portions of code in C/C++, or switching to FORTRAN compilers that do support GPU computing. Any rewriting introduces the risk of introducing new bugs, increasing the need for proper tuning, testing and software verification.

In terms of switching compilers, there are also a number of factors to consider. One of the biggest problems is the loss of productivity – possibly for a whole development team – due to changes required in build processes and utilities, the introduction of unforeseen bugs caused by incompatible compiler options, and bugs in the compilers themselves.

IV. GPU ACCELERATION IN FEKO

A. The Method of Moments

As discussed in Section III, the MoM requires the assembly and solution of a dense linear system with other steps followed like near or far field calculations. The run-time for the assembly of the matrix is quadratic in terms of the number of unknowns, whereas that of the solution of the linear system is cubic. The post-processing is typically linear in terms of the number of unknowns and linear in terms of the number of far field directions/near field observation points etc. It follows that as the problem size gets larger, the matrix solution phase will dominate the overall run-time.

The matrix solution phase can be isolated and accelerated using libraries such as MAGMA [9] or cuSOLVER (available as part of CUDA since version 7.0). Unfortunately, even though it can be accelerated by up to an order of magnitude, the total simulation acceleration is significantly less, with the matrix assembly phase now dominating the run-time. Even though considerable speedups can be attained for this matrix fill phase in simplified MoM code [3], a considerable amount of development resources need to be invested for a FEKO implementation due to the complex nature of the code (many different basis functions, higher order on curvilinear meshes, Sommerfeld integrals for planar Green's functions etc.).

B. The Finite Element Method

Another matrix-based method implemented in FEKO is the FEM. In contrast to the MoM, the matrices are sparse, but many of the same challenges present themselves when the GPU acceleration of the method is considered.

Here, the phases of the solution process which contribute most significantly to the total simulation time are the construction of the relevant preconditioner and the subsequent solution of the sparse linear system. FEKO uses by default iterative solvers for a single right hand side which – with the right preconditioners – provide according to our experience faster solution times than direct sparse solvers and in particular use less memory.

For the solution of FEM linear system, a simple iterative solver can be expected to show a 2-5x performance improvement when running on a GPU, but for most problem sizes where the amount of GPU memory is not a limitation, this translates into a simulation speedup of only 50% as the other phases start dominating.

Further acceleration is hampered by the sheer number of preconditioning options available in a software such as FEKO. In addition, differences in matrix representation and the lack of complex value support in available third-party libraries make the use of a standalone approach – as was done with the MoM matrix solution – problematic.

C. Ray launching Geometrical Optics

Along Uniform Theory of Diffraction (UTD) and Physical Optics (PO), the Ray Launching Geometrical Optics (RL-GO) solver – which is sometimes referred to as Shooting and Bouncing Rays (SBR) – is ideal for the analysis of electrically large and complex objects. It is inherently parallel and is well suited to GPU acceleration. As an initial proof of concept, we were able to accelerate the calculation of the intersections of rays with geometry in FEKO by at least an order of magnitude when using CUDA.

However, this was handwritten CUDA code. It is not possible to simply run the RL-GO code through a GPU aware compiler and obtain an accelerated implementation with similar performance. Furthermore, the complexity and recursive nature of the code means

that GPU specific limits such as smaller stack size must be addressed as well.

D. Finite Difference Time Domain Method

In much the same way that the RL-GO solver is algorithmically well suited to GPU acceleration, the Finite Difference Time Domain (FDTD) method lends itself well to such parallelization. Much of this stems from the fact that the same simple update equations are applied to each voxel in each time step with (almost) no communication required between adjacent updates. As is indicated in Fig. 1, one advantage of the acceleration of the FDTD over the RL-GO solver in FEKO is that there is as yet no hybridization of FDTD with other methods and thus, less complexity to be considered.

The FDTD solver implemented in FEKO makes used of GPU acceleration to provide roughly an order of magnitude speedup for certain problems. One disadvantage of such a speedup is that from a user's perspective, the relative performance of post-processing phases such as the calculation of far fields is significantly lower.

For both CPU and GPU based FDTD solvers, the measured performance is greatly affected by the problem setup, which includes factors such as user-requested near fields or the far fields already mentioned. If these are in the frequency domain, for example, then additional costly computations are required during every simulation time step.

V. CONCLUSION

In this paper, a discussion on the challenges associated with the GPU acceleration of the commercial CEM software package FEKO was presented. This showed that although a method may be promising theoretically, its application in commercial software generally requires the allocation of significant development resources, with at this stage not always the necessary demand from the market.

As examples, the acceleration of the MoM, FEM, and RL-GO were considered, and although certain phases of the computational process can be accelerated significantly, the total simulation speedup is limited. The further acceleration of these methods is hampered by the complexity of the numerical algorithms, e.g., through hybridization. As illustrated, for FDTD, the situation is different.

REFERENCES

[1] D. B. Davidson, *Computational Electromagnetics for RF and Microwave Engineers*, 2nd ed., Cambridge: Cambridge University Press, 2011.

[2] D. B. Kirk and W. W. Hwu, *Programming Massively Parallel Processors – A Hands-on Approach*, Burlington: Morgan Kaufmann, 2010.

[3] E. Lezar and D. B. Davidson, "GPU-accelerated methods of moments by example: monostatic scattering," *IEEE Antennas and Propagation Magazine*, vol. 52, no. 6, pp. 120-135, Dec. 2010.

[4] M. J. Inman, A. Z. Elsherbeni, and C. J. Reddy, "CUDA based GPU solvers for method of moment simulations," *Annual Review of Progress in Applied Computational Electromagnetics*, Tampere, Finland, Apr. 2010.

[5] E. Lezar and D. B. Davidson, "GPU-based Arnoldi factorization for accelerating finite element eigenanalysis," *International Conference on Electromagnetic in Advanced Applications (ICEAA)*, Torino, Italy, Sept. 2009.

[6] A. Dziekonski, A. Lamecki, and M. Mrozowski, "On fast iterative solvers with GPU acceleration for finite elements in electromagnetics," *10th International Workshop on Finite Elements for Microwave Engineering*, Mill Falls, NH, USA, Oct. 2010.

[7] K. E. Spagnoli, *An Electromagnetic Scattering Solver Utilizing Shooting and Bouncing Rays Implemented on Modern Graphics Cards*, ProQuest, 2008.

[8] Y. Tao, H. Lin, and H. Bao, "GPU-based shooting and bouncing ray method for fast RCS prediction," *IEEE Trans. Antennas Propagat.*, vol. 58, no. 2, pp. 494-502, 2010.

[9] ICL, University Tennessee, Knoxville, "MAGMA: Matrix Algebra on GPU and Multicore Architectures," 2015. [Online]. Available: http://icl.cs.utk.edu/magma/index.html

[10] E. Lezar and U. Jakobus, "GPU accelerated electromagnetic simulations with FEKO," *International Supercomputing Conference*, Hamburg, June 2012.

[11] E. Lezar, U. Jakobus, and S. Kodiyalam, "GPU related advances in the FEKO electromagnetic solution kernel," *International Conference on Electromagnetics in Advanced Applications (ICEAA)*, Torino, Italy, Sept. 2013.

[12] Altair Development S.A. (Pty) Ltd, "FEKO – Field Computations Involving Bodies of Arbitrary Shape," 2016. [Online]. Available: www.altairhyperworks.com/feko