High performance, high accuracy FDTD implementation on GPU architectures

By Guillermo Del Castillo, Chris Mason, Nicolas Chavannes, and Michal Okoniewski

1. Introduction

Simulations have become an integral part of a product design workflow, as devices are becoming more complex and development cycles are shrinking to be more competitive and meet market demand. In the last two decades, the capabilities of electromagnetic simulation techniques have vastly expanded to simulate increasingly more realistic and multifunctional devices for improved design, testing and production efficiency. Most simulation software relies on the Finite Difference Time Domain (FDTD) technique [1] for solving electromagnetic problems in a wide range of areas, including microwave and antenna engineering, bioelectromagnetics, electromagnetic compatibility, photonics, biomedicine and ultra low frequency applications. Simulations now require significant amounts of memory and processing power to generate results in a reasonable time frame. The emerging methodology applies the simulated model as the basis for an iterative design process, whereas, simulations were previously used for verification only.

In the past, computing power and long processing times ranging from hours to weeks had always been limiting factors for engineers; however, the advent of cheap supercomputing has vastly increased the computational power, which in turn has increased accuracy and speed, and allowed solutions to complex problems that previously would have been impossible. In particular, the specialized microprocessor Graphics Processing Unit (GPU) has prodigious computational capabilities optimized for 2D and 3D technology, tailor-made for the computer game and video industries. The GPU technology for High Performance Computing (HPC) architectures features massive parallel processing cores with abundant onboard memory, packaged into a footprint that fits inside a workstation. Although other platforms, including the Cell processor and the field programmable gate array (FPGA), have attempted to bring HPC technologies to the EM market, they were largely unsuccessful due to the high deployment cost and the lack of broad support. FDTD is particularly well suited to take advantage of the GPU, benefitting from a cost-effective commodity platform that provides compelling performance gains over the latest state-of-the-art workstations.

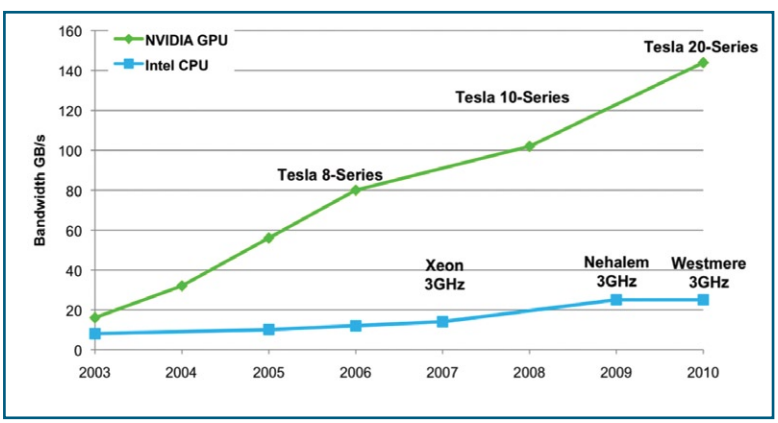

Figure 1: Comparative graph of memory bandwidth for CPU and GPU based systems.

2. Background

Schmid & Partner Engineering AG (SPEAG) [2] and Acceleware Corporation [3] joined forces in 2004 to deliver the first commercially available NVIDIA GPU accelerated electromagnetic solution using OpenGL. The SEMCAD X EM simulation platform [4] offers massive parallel processing capabilities to effectively tackle complex designs while improving product development cycles. GPUs have evolved into a highly parallelized, multithreaded platform with tremendous memory bandwidth and computational power, and have thus become a cost-effective alternative to conventional HPC systems. Refer to Figure 1.

The latest generation of GPUs from NVIDIA – Fermi – has doubled the number of cores in comparison to the previous generation (NVIDIA series 10) and features ECC (error correcting code) memory. Some applications of FDTD in which double precision is required now benefit from Fermi GPUs.

In this study we demonstrate that GPUs in general, and our implementation in particular, are most effectively suited to simulate complex, highly heterogeneous, real-world models with large domain sizes at high solver speeds and with great accuracy. SEMCAD X is the first EM simulation package to feature NVIDIA's Fermi for simulating massive models with billions of unknowns in only a few minutes.

3. Theory and methodology

A) The Compute Unified Device Architecture (CUDA) model

The introduction of the Compute Unified Device Architecture (CUDA) by NVIDIA has opened the way to high performance computing on desktop workstations. CUDA consists of the CUDA Instruction Set Architecture (ISA) and the GPU parallel compute engine. This framework provides a set of extensions to standard programming languages such as C. These extensions enable an effective implementation of parallel algorithms. However, a detailed understanding of the GPU architecture is still necessary to take full advantage of the potential performance.

On GPUs, the same program is executed for each data element and is therefore optimized for data parallelism. The memory access latency is overcome by collectively executing thousands of threads instead of utilizing large data caches. This is accomplished by the hundreds of cores in the GPU architecture. Each core has shared resources, including registers and memory. The on-chip shared memory allows parallel threads running on these cores to share data without sending it over the GPU memory bus. Threads express fine-grained data and thread parallelism whereas blocks of threads express coarse-grained data and task parallelism. Performance improves if an algorithm can exploit the use of the shared

memory, rather than relying on the slower global memory.

B) FDTD implementation in CUDA

The FDTD method belongs in the general class of the grid-based differential time-domain methods. The computational domain is discretized using Yee's cells [1]. The time evolution of the electromagnetic field is calculated in this method using central-difference approximations of the partial derivatives in the Maxwell's curl equations, whereas, the equations are solved in a leapfrog manner. When implementing FDTD to the GPU architecture, each step of the electromagnetic field update consists of:

- H-field components update
- Application of boundary conditions to the H-field
- E-field components update
- Application of boundary conditions to the E-field

Data for a field update is transferred into the shared memory to minimize overhead of communication between the global memory and the processing unit on the GPU. The FDTD method is able to take advantage of data reuse due to the spatial locality. Thread parallelization and high locality makes this implementation very efficient [5].

Not every numerical method, however, can be ported onto the GPU architecture with the same degree of acceleration. It is especially difficult to develop preconditioned sparse solvers, such as those used in Finite Element Method (FEM) that can benefit from a massively parallel architecture. Conversely, some of the characteristics of the explicit FDTD scheme make it ideal for implementation onto a graphics platform.

Recently the Khronos group introduced the Open Computing Language (OpenCL), a framework for parallel programming for mixed platforms, for example, multi-core CPUs, GPUs, and other modern processors. This standard opens the way to build so-called heterogeneous computing systems, which may simultaneously deploy the computational power of multi-core CPUs and GPUs. However, with currently available versions, high performance can only be obtained after specific and extensive code optimizations tailored to the architecture [7]. As of this writing, the OpenCL FDTD simulations still perform at a lower speed than native CUDA or OpenMP (CPU-Based) implementations. Yet, it is likely that that the OpenCL framework will gain large popularity in the coming years and might become the standard with respect to parallel programming.

C) Methods

The simulation platform SEMCAD X V14.2 [4] was applied for all implementations and assessments presented in this paper. SEMCAD X is a universal simulation platform with a high-end ACIS® based modeler, CAD importer and graphical user interface (in-house 3-D OpenGL renderer) that integrates various solvers providing native 64 bit functionality, such as full-wave EM solvers (FIT/C-FDTD, C-ADI-FDTD, etc.), FEM based low frequency and static solvers, thermal solvers, coupled full-wave EM-SPICE circuit solvers and a GA based optimization platform running in parallel on a distributed network. Since its inception in 2000, Acceleware has continued to evolve its acceleration libraries to meet the rapidly changing landscape of high performance computing by adding support for NVIDIA's CUDA architecture, support for multiple GPUs in a system, and support for clusters of GPUs.

High performance GPUs can be added to any conventional computer system. A candidate computer needs at least one PCI-Express slot to use a GPU accelerator card since a high-bandwidth data bus is required.

Depending on the desired computing power and number of GPUs, a NVIDIA Tesla C1060/C2050 (card), QuadroPlex D2 (substation) or Tesla S1070/S2050 (1U form factor) can be added to a personal computer. These devices have 1 GPU, 2 GPUs and 4 GPUs, respectively. Each GPU has 3-4 GB of RAM.

In addition, for large domains that do not fit in a single GPU device or that require faster solver speeds than an out-of-the-box device, it is possible to build a cluster of GPUs. These GPU-clusters use the same paradigm of a Beowulf cluster of CPUs: a single head node where the jobs are delivered to and a series of compute nodes with GPU devices attached to them. Domain decomposition is performed via MPI in an analog fashion of a Beowulf cluster. The GPU RAM of the compute nodes shadows the head node CPU memory. These systems can be built up to 16 compute nodes. Infiniband and Ethernet network protocols provide the fast connection between the different elements. The following section describes some of the benchmark examples that were executed in one of these cluster-GPU systems.

4. Discussion and results

A) Benchmark outline

Three different benchmarks were executed using a special array of clustered GPUs. They were selected to outline characteristics that would be unreasonably difficult or impossible to be performed in a CPU-based system. Moreover, they focus on very topical fields within today's industrial application range of EM simulations.

A.1)

The first example targets antenna design and compliance, highlighting a large domain with a big spatial extension, small embedded details and high inhomogeneity. It consists of a domain of more than 500 Mcells in size, limited by 8 UPML layers. A full CAD model of a car with a driver (non-homogeneous posable anatomical CAD based full-body model [6]) holding a CAD derived cell phone was used (Figure 2). The phone is modeled with details as low as 10 microns. The antenna in this device provides an excitation at 1800 MHz. The number of individual CAD parts is beyond 2000. All performance results and comparisons are discussed at the end of this section.

A.2)

The second example targets medical devices and compliance of a human head with an implant made of an extremely fine coil of approximately 0.07 mm radius, requiring great detail and resolution. Currently, there is increasing interest to perform numerical evaluations and optimizations of implantable medical devices in an industrial environment. The practical challenges of solving this particular problem involve a very finely detailed, relatively long helicoidal structure. In addition, the birdcage resonator of a large magnetic field in which the head is immersed, requires a large domain. This setup results in a large size simulation with a very fine grid.

The second benchmark has a domain of about 350 Mcells in size, bound by 11 UPML layers. A human male head [6] model with an implant for deep brain stimulation and a box

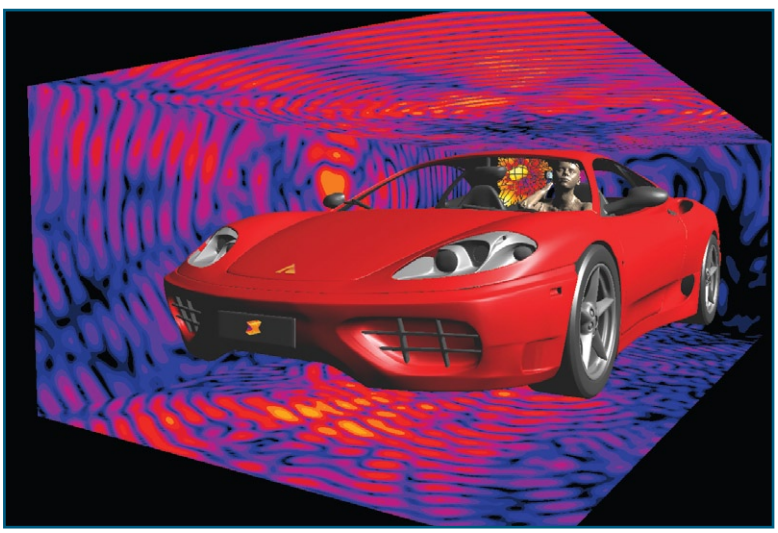

Figure 2: The Benchmark 1 model consists of a car, driver and cell phone. In this picture, the electric fields of the simulation have been overlaid to the model.

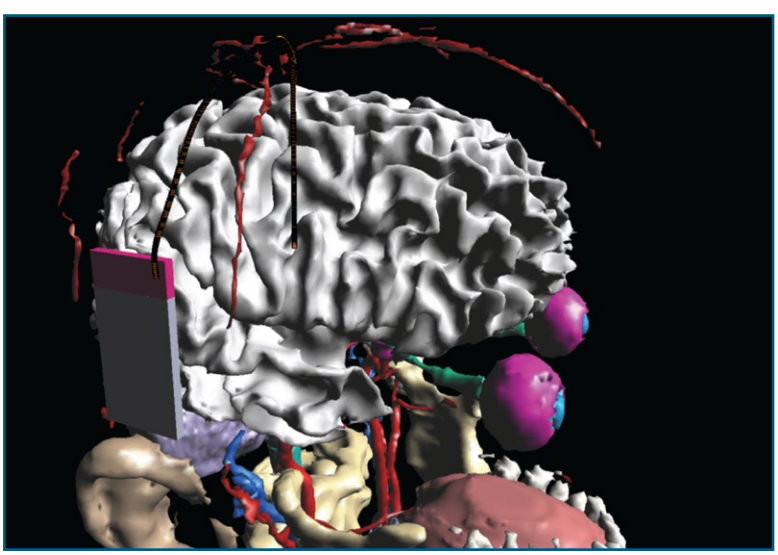

Figure 3: The Benchmark 2 model includes a human male head, deep brain stimulation implant and box.

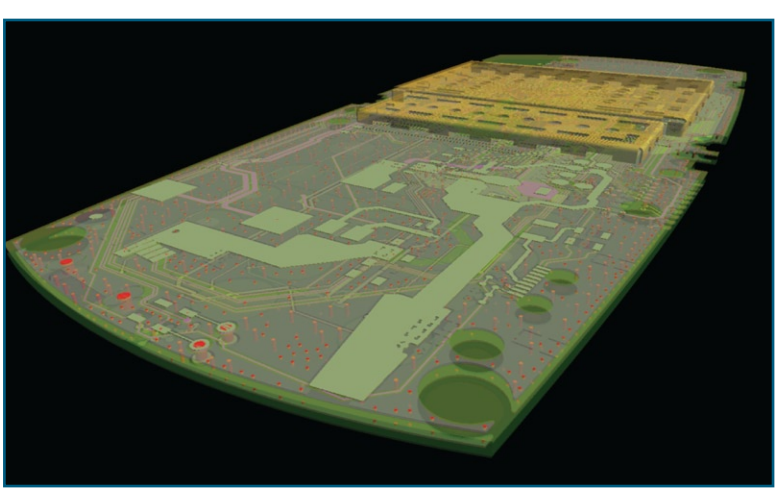

Figure 4: The Benchmark 3 model is a full CAD PCB board.

was used. The implant consists of a metallic stimulator and a helicoidal wire insulated with a sleeve along its length, ending in an uncapped 10 mm cylindrical tip at the distal end. The implanted head is exposed to the RF fields of a 1.5 T birdcage coil. This device provides the excitation at 64 MHz. Refer to Figure 3. The performance of the implemented GPU system for this example is outlined at the end of the section.

A.3)

The third proposed problem targets the investigation and optimization of the EMI/EMC behavior of today's PCBs within fields of communication and computing. This benchmark consists of a domain of more than 370 Mcells in size, bound by 9 UPML layers. The simulation addresses a detailed characterization of an industrial multilayer PCB with particular focus on the improvement of shielding and cross-talk effects by different shield types. The model consists of several 1000's of distinguished CAD parts modeled at resolutions as low as 1 micron as well as numerous lumped elements. A broadband pulse (0.5 – 2 GHz) was applied for the excitation of the various assessments. Refer to Figure 4.

AB Performance Results

Table 1 incorporates the specifications for each benchmark. The cluster GPU-array used to calculate the proposed examples consists of one head node and three compute nodes running Red Hat Linux 64 bit. The compute nodes are HPxw9400 workstations with Dual Core AMD Opteron 2216 at 2.4 GHz and 16 GB of RAM. Each of these nodes is connected to an NVIDIA Tesla S1070 1U chassis. The head node has 64 GB RAM, and each compute node has 24 GB RAM and 16 GB GPU RAM. The nodes are connected with a common file system over Infiniband. The array uses MPI over Ethernet for domain decomposition. The FDTD solver is the SEMCAD X Linux Solver using the Acceleware Cluster Libraries. For comparison, these examples were also executed with the standard, CPU-based, SEMCAD X software solver for comparison. An HPxw9400 workstation with Dual Core AMD Opteron 2216 at 2.4 GHz and 16 GB of RAM using Windows 7 64 bit as the operating system was utilized. Table 2 outlines the solver speeds and computation times.

The GPU results were generated in a cluster comprising 12 GPUs (3 nodes x 4 GPU in a Tesla S1070) versus the two processing units used by the CPU solver. The performance is outstanding. The cluster solver runs more than 120 times faster than the speed of the CPU solver. Simulations that were previously impossible to perform within a reasonable time or that required tedious simplifications of the model or cropping of the domain can now be run "as is" and executed in just minutes.

The flexibility and power of using a GPU-based architecture makes it possible to run large domain, realistic models with great accuracy without having to sacrifice detail or time. Figure 2 shows the results of the full EM simulation overlaid onto the model of the car and driver. Due to the sheer size of the example, Figure 5 highlights the area of interest of the electrical field near the cell phone and hand in the same initial benchmark. This entire example can be computed in less than 30 minutes using SPEAG's SEMCAD X with Acceleware's libraries.

One final consideration concerned the running of the first presented example using the latest NVIDIA GPU technology. This test offers a preview of the performance of the new state of the art NVIDIA series 20 (Fermi). The benchmark was run on a single HPxw9400 workstation using a NVIDIA Tesla C1060, a NVIDIA Fermi C2050 and Acceleware's Nehalem software solver. The simulation sizes are smaller compared to the previous tests since the simulations were performed on a single machine and not a GPU-clustered system. The RAM of the system (or conversely the RAM of the GPU) limits the maximum domain sizes. The results of this test are shown in Figure 6.

Both the C2050 card and the M2050 use NVIDIA's Fermi architecture. This architecture adds improved double precision

Table 1: The three benchmark simulation setups.

SEMCAD X Simulation	Benchmark 1	Benchmark 2	Benchmark 3
Model	car, driver, phone	head with implant	PCB
Frequency (MHz)	1800	64	500–2000
No. time step	5896	406435	404530
Computational domain (Mcell)	5037	3558	3743
Excitation	antenna	resonator	coax line
ABC	UPML 8 lyrs	UPML 11 lyrs	UPML 9 lyrs

Table 2: The simulation speeds and times of all three examples.

Solver Performance	Benchmark 1	Benchmark 2	Benchmark 3
GPU Solver Speed (Mcell/s)	2420	2200	653
GPU Solver Time (hh:mm)	0:29	21:10	06:19
CPU Solver Speed (Mcell/s)	27	198	156
CPU Solver Time (hh:mm)	60:21	2043	292

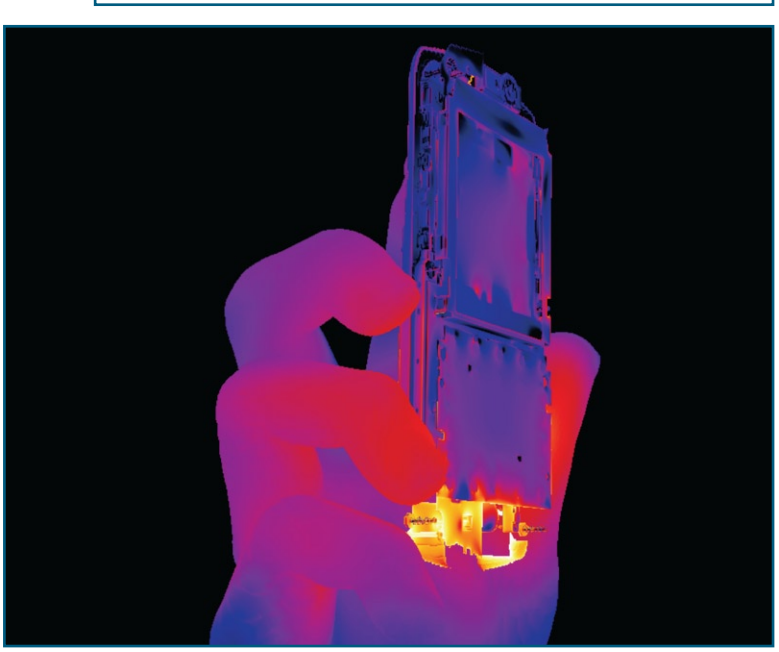

Figure 5: Electric field in phone and hand of benchmark 1 (detail).

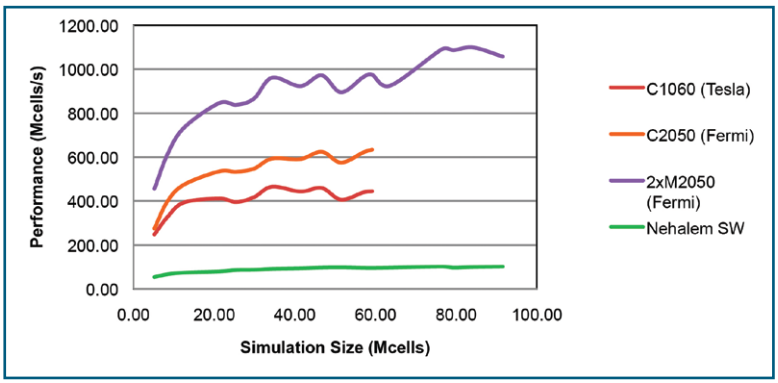

Figure 6: Performance results for latest generation GPUs using the setup of benchmark 1.

throughput, ECC support, and improved overall performance. In addition, two of these cards can be connected in parallel to allow higher performance and larger domain sizes. The Series 20 shows remarkable improvement over the Series 10 (Tesla) in the above plot, but it positively eclipses the solver speed of the Nehalem software solver. Solving this particular problem in CPU-based software mode simply becomes too onerous to be practical.

5. Price/performance considerations

The GPU is a cost effective solution when compared to an equivalent CPU cluster in terms of performance. Assuming an approximate cost of $3500 per CPU node including switches, achieving a performance of 450 MCells/s would require 10 CPU nodes (~$35000) compared to a single Fermi (MSRP $2500) plus a workstation ($3500). The price performance ratio is improved by approximately 6x when using a GPU. This does not include the recurring annual cost of power, cooling, and administration for a CPU cluster..

6. Conclusions

In this article we demonstrated the improved computational power of specialized High Performance Computing (HPC) EM simulation systems derived from the latest generation of GPUs based on the simulation platform SEMCAD X and Acceleware's acceleration libraries.

The FDTD method is particularly suited to be parallelized using the high-thread count, high memory transfer characteristic of GPU-based solutions, offering high performance and high solver speeds at a relatively low price. Prodigious computational power is no longer exclusive to large clusters and supercomputers as GPUs are readily available for the desktop computer of an RF engineer. Massively large simulations can be performed using a cluster of GPUs with the Beowulf cluster model as a template. This setup increases the size and the performance of the system by distributing the computational load using MPI.

In addition, new frameworks like OpenCL – which can be executed in so-called heterogeneous platforms – can potentially become the new standard in parallel programming and replace CUDA and OpenMP in the long term.

The performance of our implementation was demonstrated in three specific high-end simulation examples. Simulation speedups of approximately 150x can be achieved at a relatively low cost, allowing the computation of highly complex real-world scenarios in less than 30 minutes as compared to several days for CPUs. In addition, this joint study shows the initial results of the first commercial implementation and provision of the new NVIDIA Fermi devices, indicating a significant technological jump over its predecessor, the NVIDIA Tesla series.

References

[1] A. Taflove and S.C. Hagness, "Computational Electrodynamics: The Finite-Difference Time-Domain Method", 3rd. ed., USA, Artech House Publishers, 2005.

[2] Schmid & Partner Engineering AG, www.speag.com.

[3] Acceleware Corporation, www.acceleware.com.

[4] SEMCAD X Reference Guide, V14.2, www.semcad.com.

[5] Krakiwsky, S. E., , L. E. Turner, and M. M. Okoniewski, "Acceleration of finite-difference time-domain (FDTD) using graphics processor units (GPU)," IEEE MTT-S Int. Microwave Symp. Digest, 1033-1036, 2004.

[6] Andreas Christ et al, "The Virtual Family—development of surface-based anatomical models of two adults and two children for dosimetric simulations", 2010 Phys. Med. Biol. 55 N23.4.

[7] T. P. Stefański, S. Benkler, N. Chavannes and N. Kuster, "Parallel Implementation of the Finite-Difference Time-Domain Method in the Open Computing Language", Proceedings of the ICEAA, 2010.

Authors

The authors, Guillermo Del Castillo and Nicolas Chavannes work for Schmid & Partner Engineering AG (SPEAG), Zurich, Switzerland, while Chris Mason and Michal Okoniewski contributed on behalf of Acceleware Corporation (AXE), Calgary, Canada.

Acknowledgments

This study was generously supported by the Swiss Commission for Technology and Innovation (CTI) and assisted by the Foundation for Research on Information Technologies in Society (IT'IS), Switzerland.