Hardware Acceleration of Electromagnetic Field Profile Computation: A Case Study Using the PO-SBR Method

Eric Dunn[1], Nathan Smith[1], Ray Hoare[2], Huan-Ting Meng[3], and Jianming Jin[3]
[1]Science Applications International Corporation ({eric.a.dunn, nathan.j.smith}@saic.com)
[2]Concurrent EDA (rayhoare@concurrenteda.com)
[3]University of Illinois at Urbana-Champaign ({meng2, j-jin1}@illinois.edu)

Introduction

Many defense and commercial communication system applications rely on electromagnetic simulations for signal exploitation and mission planning. For instance, when deploying a communication system it is important to predict the coverage that the system will provide. Such prediction often takes the form of field profiles, which are color plots indicating the simulated field strength from a transmitter over a geographic area. These plots are frequently overlaid on maps to allow planners to ensure, for example, that soldiers can maintain a communication link with command in the battlefield or cellular providers can ensure their cell phone users do not experience dropped calls.

Despite their importance, field profiles can be difficult to generate. Physical measurements take time and often are impossible if access to the area is restricted. For numerical simulation based on computer models, even the fastest algorithms often take significant time to carry out the computation. This is because field values must be found not only for a large number of observation points, but also for multiple frequencies.

In order to provide planners with this critical information, the computation time required to generate field profiles should be greatly reduced, ideally to real-time speeds. If such a reduction were possible, transmitters and receivers could be relocated and still operate as originally intended in response to changes in the operating environment.

Fortunately, with the right choice of simulation algorithm, the field value at each observation point/frequency pair can be computed independently of all other observation point/frequency pairs. This suggests that significant performance increases can be realized using the increasingly parallel processing capabilities afforded by multi-core Central Processing Units (CPUs), Graphics Processing Units (GPUs), and even Field Programmable Gate Arrays (FPGAs).

In this paper we present a case study comparing such a field-profile generation algorithm implemented as a GPU code with the same algorithm implemented as a CPU code. We also present the predicted performance when the algorithm is implemented on an FPGA as estimated by Concurrent Analytics [1]. Previous research in [2] has looked at GPU acceleration of Radar Cross Section calculation, but to our knowledge this is the first time the use of a GPU has been investigated for calculating very large field profiles. The combination of fast hardware along with properly chosen simulation algorithms will be shown to bring this military and commercial application a step closer to becoming real-time embedded computing.

Simulation Algorithm

There are many algorithms available for the simulation of electromagnetic radiation and scattering. Here we chose to use what is known as the Physical Optics – Shooting and Bouncing Ray (PO-SBR) algorithm [3]. This algorithm involves two separate processes.

In the first stage, known as SBR, a CAD model is loaded and used to represent the environment in which the fields will be measured. A bundle of rays are launched radially from each transmitter. Each generated ray is traced through the scene and is reflected off of each surface of the environment it encounters.

In the second stage, known as PO, each of the ray/surface interactions gets mapped to an equivalent current. Those currents then radiate to each field observation point as a function of the signal frequency, surface normal and distance from the ray/surface interaction point to the observation point.

Hardware Acceleration

There are two types of hardware we investigate in this paper. First we compare a CPU implementation of the PO-SBR algorithm to a GPU implementation of the same algorithm. The CPU implementation is single-threaded, despite the use of an Intel® Xeon® E5530 processor, which contains four cores. While this processor is capable of executing four threads concurrently, the CPU metrics only consider serial execution and serve as a baseline. This implementation uses the well-known PBRT ray-tracer to perform the SBR stage [4].

The GPU implementation was developed using CUDA^{TM}, NVIDIA®'s GPGPU language. All benchmarks were taken on an NVIDIA® Quadro® 5800 card, which contains 4GB of global memory and 240 thread processors. Each stage of the algorithm is parallelized differently according to its requirements.

The SBR stage is parallelized using NVIDIA®'s OptiX^{TM} package since each generated ray is independent of others. The PO stage is parallelized across observation points and frequencies. Each thread on the GPU will loop over all ray/surface interactions from the current ray-trace and radiate the assigned current from the intersection point to the observation point, where the field is accumulated.

The potential performance of the algorithm on an FPGA is also considered. For this we used the Concurrent Analytics tool developed by Concurrent EDA. This tool instruments and observes the application as it executes.

An instruction-level analysis is then further exploited for loop-level and instruction-level parallelism that is eligible to be converted into FPGA operations. The output is FPGA area and performance estimation.

Results

To compare the different hardware acceleration methods we chose the example problem shown in Figure 1. This geometry consists of a collection of four metal plates distributed around a transmitter. In this case the transmitter represents an electric Hertzian dipole antenna with a center frequency of 10GHz. At this frequency the total area the fields are computed over is 16m by 18m.

The reason we chose this example is because it gave us a controlled environment that would still be challenging for most simulation software. What makes this a challenging problem is the size of the geometry involved and the multiple paths that energy can propagate over – thus requiring an electromagnetic simulation for an accurate prediction of the field strength.

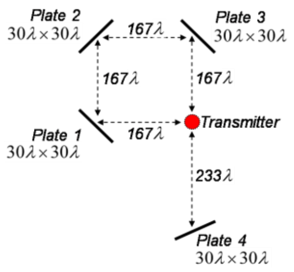
Plate 2
$30\lambda \times 30\lambda$
167λ
Plate 3
$30\lambda \times 30\lambda$
167λ
167λ
Plate 1
$30\lambda \times 30\lambda$
167λ
Transmitter
233λ
Plate 4
$30\lambda \times 30\lambda$

Figure 1: Geometry consisting of four plates.

To produce the field profiles shown in Figure 2 the ray tracing part of the PO-SBR simulation consisted of 12,746 intersection points that were then radiated to different observation points. In these plots the red color indicates stronger field strength and the blue color indicates weaker field strength. For instance, the blue regions are where the field is shielded by the metal plates.

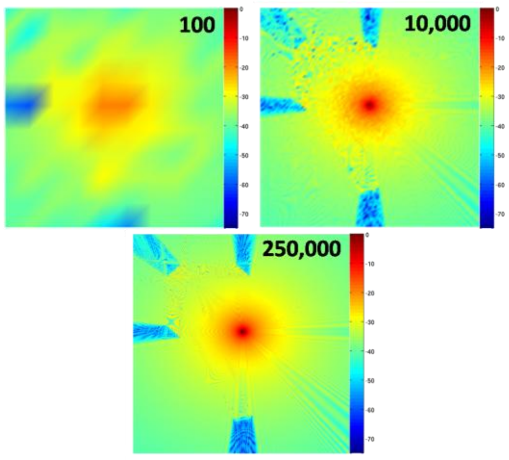

Figure 2: Field profiles with 100, 10,000 and 250,000 observation points.

Shown in Figure 3 is a plot of the performance speed-up for both GPU and FPGA accelerated versions.

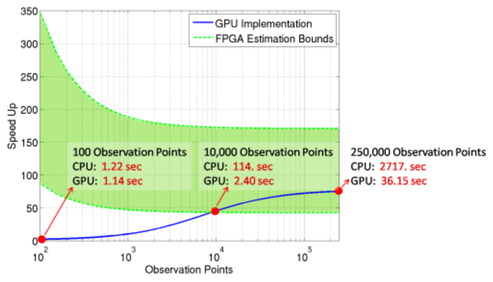

Figure 3: Speed up of device run time relative to CPU.

The FPGA analysis was focused on the PO portion of the application as this consumes almost all of the total application time. The inner loop of PO was determined to be 1130 processor instructions of which 131 were floating point operations and seven were math library calls. A pipeline of FPGA operations was constructed for analysis that did not include the systems calls and resulted in 275 stages. It is estimated that the system calls will add another 50-100 cycles. Concurrent EDA's tools guarantee 200MHz performance and thus the first results will require at least 1625ns while each subsequent result will only require 5ns since each pipeline stage can operate on a different combination of hit point, observation point, and frequency. Thus, the steady state performance is one result every 5ns. Latency from a host processor to an in-socket FPGA is under 100ns and can also be pipelined. Expected steady state performance is between 5ns and 20ns per combination for 1,000 combinations and higher.

To put these numbers in context for this case study, to obtain a field profile with a resolution of one square-wavelength over a 16m by 18m area due to a 10GHz transmitter a CPU code would require about 45min. But using a GPU we are able to compute the same field profile in only about 35sec. Using the FPGA estimations, the expected performance on an FPGA could be reduced to 16sec.

Conclusion

This case study has shown that hardware acceleration, such as that available with current GPU and FPGA technology, can significantly reduce the simulation time required for calculating the field profile generated by a PO-SBR simulation. Having the ability to rapidly and accurately calculate these field profiles can improve many military and commercial communication planning and operations.

References

[1] Concurrent EDA, *Concurrent Analytics.* http://www.concurrenteda.com/analytics.php.

[2] Y. Tabo, H. Lin, and H. Bao, "GPU-Based Shooting and Bouncing Ray Method for Fast RCS Prediction," *IEEE Trans. Antennas & Propagat.,* Vol. 58, No. 2, Feb. 2010.

[3] H. Ling, R. C. Chou, and S. W. Lee, "Shooting and Bouncing Rays: Calculating the RCS of an Arbitrary Shaped Cavity," *IEEE Trans. Antennas & Propagat.,* Vol. 37, No. 2, Feb. 1989.

[4] M. Pharr and G. Humphreys, *Physically Based Rendering from Theory to Implementation*. Morgan Kaufmann, 2004. http://www.pbrt.org.