Programming Finite-Difference Time-Domain for Graphics Processor Units Using Compute Unified Device Architecture

Veysel Demir* (1) and Atef Z. Elsherbeni (2)
(1) Department of Electrical Engineering, Northern Illinois University, USA
(2) Department of Electrical Engineering, University of Mississippi, USA
E-mail: demir@ceet.niu.edu, atef@olemiss.edu

Abstract

Recently graphic processing units (GPU's) have become the hardware platforms to perform high performance scientific computing them. The unavailability of high level languages to program graphics cards had prevented the widespread use of GPUs. Relatively recently Compute Unified Device Architecture (CUDA) development environment has been introduced by NVIDIA and made GPU programming much easier. This contribution presents an implementation of finite-difference time-domain (FDTD) method using CUDA. A thread-to-cell mapping algorithm is presented and performance of this algorithm is provided.

Introduction

Recent developments in the design of graphic processing units (GPU's) have been occurring at a much greater pace than with central processor units (CPU's). The computation power due to the parallelism provided by the graphics cards got the attention of communities dealing with high performance scientific computing. The computational electromagnetics community as well has started to utilize the computational power of graphics cards for computing and, in particular, several implementations of finite-difference time-domain (FDTD) method have been reported. Initially high level programming languages were not conveniently available to program graphics cards. For instance, some implementations of FDTD were based on OpenGL. Then Brook [1] has been introduced as a high level language for general programming environments, and for instance used in [2] as the programming language for FDTD. Moreover, use of High Level Shader Language (HLSL) as well is reported for coding FDTD. Relatively recently, introduction of the Compute Unified Device Architecture (CUDA) [3] development environment from NVIDIA made GPU computing much easier.

CUDA has been reported as the programming environment for implementation of FDTD in [4]-[7]. In [4] the use of CUDA for two-dimensional FDTD is presented, and its use for three-dimensional FDTD implementations is proposed. The importance of coalesced memory access and efficient use of shared memory is addressed without sufficient details. Another two-dimensional FDTD implementation using CUDA has been reported in [5] however no implementation details are provided. Some methods to improve the efficiency of FDTD using

978-1-4244-4968-2/10/$25.00 ©2010 IEEE