GPU Based Acceleration of Telegraph equation

Václav Šimek, Michal Kraus, Jiří Kunovský, Jiří Petřek,
Brno University of Technology
Faculty of Information Technology
Božetechova 2, 612 66 Brno, Czech Republic
simekv@fit.vutbr.cz (Václav Šimek)

Abstract

In a matter of just a few years, the programmable graphics processor unit has evolved into an absolute computing workhorse. With multiple cores driven by very high memory bandwidth, today's GPUs offer incredible resources for both graphics and non-graphics processing.

An original mathematical method "Modern Taylor Series Method" (MTSM) which uses the Taylor series method for solving differential equations in a non-traditional way has been developed and implemented in TKSL software [3]. Even though this method is not much preferred in the literature, experimental calculations have shown and theoretical analyses have verified that the accuracy and stability of the Taylor series method exceeds the currently used algorithms for numerically solving differential equations. It is the aim of the paper to illustrate GPU and MTSM for numerical solutions of a telegraph line.

1. Introduction

The eletricial problem - *Telegraph equation* models the behaviour of eletrical signals on a telegraph line. The first step in building the model is to replace a small piece of the wire by quadrupole build from resistors, capacitors and coils Figure 1. Letting the size of this piece go to zero we obtain a partial differential equation describing the behaviour of signals on the wire Eq. (1)

$$\begin{aligned} L\cdot C\frac{\partial^2 u(x,t)}{\partial t^2} + (L\cdot G + C\cdot R)\frac{\partial u(x,t)}{\partial t} + \\ +R\cdot G\cdot u(x,t) - \frac{\partial^2 u(x,t)}{\partial x^2} &= 0 \\ L\cdot C\frac{\partial^2 i(x,t)}{\partial t^2} + (L\cdot G + C\cdot R)\frac{\partial i(x,t)}{\partial t} + \\ +R\cdot G\cdot i(x,t) - \frac{\partial^2 i(x,t)}{\partial x^2} &= 0 \end{aligned} \quad (1)$$

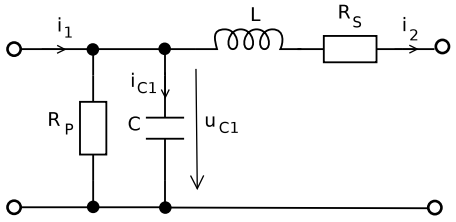

Figure 1. Modelling a small piece of the wire

One small piece of the wire can be decribed by equations that follow:

$$\begin{aligned} u'_{C1} &= \frac{1}{C_1}\cdot i_{C1} \qquad u_{C1}(0) = 0 \\ i'_2 &= \frac{1}{L}\cdot(u_{C1} - R_S * i_2) \quad i_2(0) = 0 \end{aligned}$$

2. Modern Taylor Series Method

The main idea behind the Modern Taylor Series Method is an automatic integration method order setting, i.e. using as many Taylor series terms for computing as needed to achieve the required accuracy.

Simulation of the telegraph line is presented in Figure 2.

Telegraph line has been modelling using small pieces of the wire. As an example 10 segments from Figure 1 have been used to show expected time functions (input function $U1$, output function $U2$ and ORD - order of the Taylor Series Method used) of the telegraph line (Figure 2).

Experimental calculations can be done theoretically with an arbitrary integration step. For integration step $h = 10^{-11}$ (in Figure 2) TKSL automatically sets the order between 4–11.

A lot of experiments and calculations can be done with the model of telegraph line.

978-0-7695-3114-4/08 $25.00 © 2008 IEEE
DOI 10.1109/UKSIM.2008.47

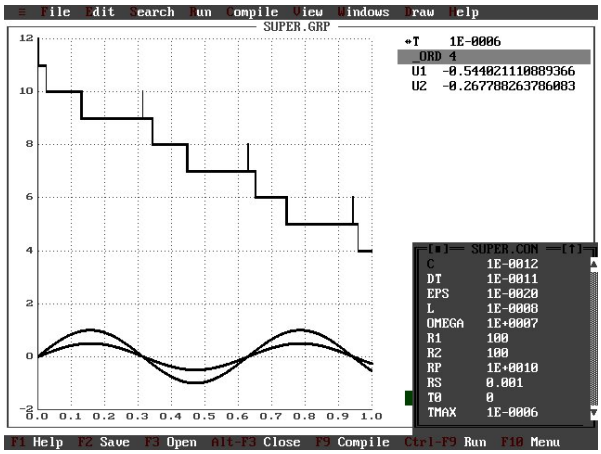

Figure 2. Time functions U1, U2, ORD

3. The Graphics Processor Unit as a Data-Parallel Computing Device

It can be expected that number of segments of the telegraph line may significantly influence the computation time. That is why, the use of GPU will be analyzed in our next experiments.

The GPU is especially well-suited to address problems that can be expressed as data-parallel computations with high arithmetic intensity. Because the same program is executed for each data element, there is a lower requirement for sophisticated flow control; and because it is executed on many data elements and has high arithmetic intensity, the memory access latency can be hidden with calculations instead of big data caches

Data-parallel processing maps data elements to parallel processing threads. Many applications that process large data sets such as arrays can use a data-parallel programming model to speed up the computations.In fact, many algorithms are accelerated by data-parallel processing, from general signal processing to physics simulation or to computational biology.

Up until now, however, accessing all that computational power packed into the GPU and efficiently leveraging it for non-graphics applications remained tricky:

The GPU could only be programmed through a graphics API, imposing a high learning curve to the novice and the overhead of an inadequate API to the non-graphics application.

The GPU DRAM could be read in a general way - GPU programs can gather data elements from any part of DRAM - but could not be written in a general way - GPU programs cannot scatter information to any part of DRAM -, removing a lot of the programming flexibility readily available on the CPU.

Some applications were bottlenecked by the DRAM memory bandwidth, under-utilizing the GPU's computational power.

This paper describes a novel hardware and programming model and exposes the GPU as a truly data-parallel computing device.

CUDA (Compute Unified Device Architecture) [1] is a new hardware and software architecture for issuing and managing computations on the GPU as a data-parallel computing device without the need of mapping them to a graphics API.

The CUDA software stack is composed of several layers: a hardware driver, an application programming interface (API) and its runtime, and two higher-level mathematical libraries of common usage, CUFFT and CUBLAS. The hardware has been designed to support lightweight driver and runtime layers, resulting in high performance.

When programmed through CUDA, the GPU is viewed as a compute device capable of executing a very high number of threads in parallel [2]. It operates as a coprocessor to the main CPU, or host: In other words, data-parallel, compute-intensive portions of applications running on the host are off-loaded onto the device.

4. Summary

Experimental calculations have shown and theoretical analyses have verified that the accuracy and stability of the Taylor series method exceeds the currently used algorithms for numerically solving differential equations. Calculations can be done theoretically with arbitrary integration step.

All experiments have been done with TKSL software. It can be expected that number of segments of the telegraph line may significantly influence the computation time. That is why, the use of GPU will be analyzed in our next experiments.

Details will be presented in the conference.

References

[1] *NVIDIA CUDA Compute Unified Device Architecture*, 2007. Programming Guide.

[2] S. Che, J. Meng, J. W. Sheaffer, and K. Skadron. A performance study of general purpose applications on graphics processors. In *First Workshop on General Purpose Processing on Graphics Processing Units*, page 10.

[3] J. Kunovský. *Modern Taylor Series Method*. FEI-VUT Brno, 1994. Habilitation work.