GPU-based Parallelization of System Modeling

S. Pachnicke
ADVA Optical Networking SE, Maerzenquelle 1-3, 98617 Meiningen, Germany
spachnicke@advaoptical.com

Abstract: Simulation of fiber-optic transmission systems using general purpose-computing on graphics processing units is investigated. It is shown that a speedup factor of more than 100 can be achieved compared to CPUs without loss in accuracy.

OCIS codes: (060.2330) Fiber optics communications; (220.4830) Systems design

1. Introduction

Modeling and simulation play a major role in research and development of optical transmission systems. Optimum design of such systems is crucial to make best use of optical transmission equipment which requires very high capital expenditure. Especially accurate modeling of the fiber is important as it is the key component of optical transmission networks. Propagation of light in a single-mode glass fiber, which is used almost exclusively for long-distance communications today, is described by the nonlinear Schrödinger equation, which is solved in state-of-the-art transmission system simulators by the split-step Fourier method (SSFM) for arbitrary input conditions. Unfortunately, even on today’s computer systems the simulation of a wavelength division multiplex (WDM) transmission system with a high channel count, long bit sequences and a transmission distance of several thousand kilometers is still very time-consuming with computational efforts of many hours to several days.

This is why already more than a decade ago parallelized implementations of the SSFM have been investigated [1]. At that time only supercomputers made possible the use of several processors in parallel with access to a shared memory. Since then the development of central processing units (CPUs) progressed rapidly with year-over-year increasing performance values (usually measured in floating-point operations per second, Flops). Nowadays also CPUs of desktop or notebook PCs usually comprise several cores. For several years, however, the performance of graphics processing units (GPUs) has been increasing much faster than estimated by Moore’s law [2] (compare Fig. 1) mainly to meet the requirements of computer games and entertainment applications [3]. Today GPUs by far exceed the computational power of CPUs. This high performance makes GPUs very attractive as co-processors for general-purpose computation. Current graphics cards offer a high number of processing cores at a reasonable cost and are already available in many standard desktop PCs, which are typically used by a system designer.

In this paper the benefit of using GPUs in modeling of fiber-optic transmission systems is quantified. As many graphics cards show a significantly higher performance in single precision (SP) arithmetic (because the main target group is computer gamers, where high accuracy is not key) also SP-based algorithms are investigated regarding speedup and accuracy, and methods for increasing the accuracy of the algorithms are discussed.

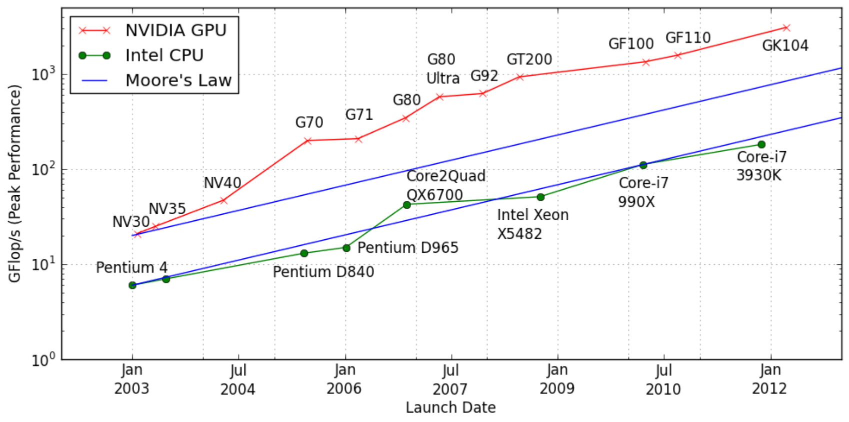

Fig. 1. Development of the computational performance (measured in GFlop/s) for selected CPUs and GPUs. Note that peak performance of a GPU is achieved only in single precision arithmetic.

2. Parallelized Implementation of the Split-Step Fourier Method

The SSFM, which is used to solve the nonlinear Schrödinger equation (NLSE), computes the linear part of the

NLSE in frequency domain and the nonlinear part in time domain [4]. As both parts are treated independently only small step-sizes are admissible to keep the error low, which results from insufficient consideration of the interaction between the linear and nonlinear operators. For each split-step a fast Fourier transform (FFT) and an inverse FFT are required for conversions from time to frequency domain and vice versa. The total computational time for a typical system simulation with non-negligible fiber nonlinearity is thus dominated by the calculation of FFTs (and IFFTs, respectively). The implementation of discrete and fast Fourier transforms on a GPU has been studied extensively in recent years (compare e.g. [5]). FFT routines are available in the form of pre-compiled libraries for GPUs (such as the CUFFT library by NVIDIA). The performance of such algorithms on a GPU, however, strongly depends on the hardware architecture and the clock rate of the different memories and caches involved [6]. This is why an auto-tuning algorithm – automatically selecting the optimum factorization (choice of radix) for a certain FFT length and accounting for the characteristics of a specific graphics card – is advantageous (as explained in [7], [8]).

Another important aspect is the accuracy of the results. The accuracy of an FFT can be measured by assessing the backward error, which is calculated by performing an FFT and IFFT operation for an arbitrary input sequence and comparing the results to the initial sequence. For double precision (DP) arithmetic the backward error (rmse) lies in the range of 10^{-7}, independently of the execution on a CPU or GPU [7], and both are usually compliant to the IEEE standard for floating point arithmetic (IEEE 754-1985). In the case of single precision arithmetic the backward error naturally increases significantly (compare Fig. 2, left). Furthermore it can be observed that the backward error strongly depends on the actual implementation. The reason for this behavior mainly lies in the dependence of the FFT on the accuracy of the trigonometric function values used in the so-called "twiddle-factors" [9]. Significant improvements in accuracy can be gained by calculating the twiddle-factors in double precision (and converting to single precision afterwards). As for an FFT of a given size the values of the trigonometric functions are known, and in the SSFM typically thousands of FFTs of the same size are executed, it is most efficient to store the twiddle-factors in a look-up table that needs to be calculated only once. The backward errors for two GPU- and two CPU-based FFT implementations - all using single precision arithmetic (including the widely used "Fastest Fourier Transform in the West" (FFTW) developed by the MIT [10]) - are shown in Fig. 2 (left). It can be observed that by a look-up table based approach the backward error of the FFT algorithm on the GPU can be decreased significantly and lies even below the CPU-based FFTW.

The implementation of the entire SSFM on a GPU has been presented first in [11] based on the NVIDIA CUDA library. It has been shown that results from GPU-based simulations in double precision have negligible deviation from simulation results from a CPU. In double precision a speedup factor of up to 50 can be achieved comparing GPU and CPU simulation times (using a single CPU core only). In single precision an even higher speedup of more than 140 can be reached (compare Fig. 2, right). The increased performance for a higher FFT length can be attributed to a better utilization of the GPU hardware. Furthermore the transfer between main memory and graphics memory is dominating the calculation time for smaller FFT sizes.

For bit error rate (BER) estimations with a pre-defined confidence level often Monte Carlo (MC) simulations are used. In the literature several improvements for MC have been presented among them "stratified sampling", which is an adapted form of importance sampling [12]. Stratified sampling partitions the sample space by fast approximates of the real system performance. This approximate solution can be obtained in this case by single precision simulations. The amount of required simulations in double precision is then determined algorithmically for meeting a pre-defined confidence level of the results (e.g. BER level at the receiver). It has been shown that this approach allows a speedup of up to 180 with accuracy comparable to CPU-based simulations in double precision [8].

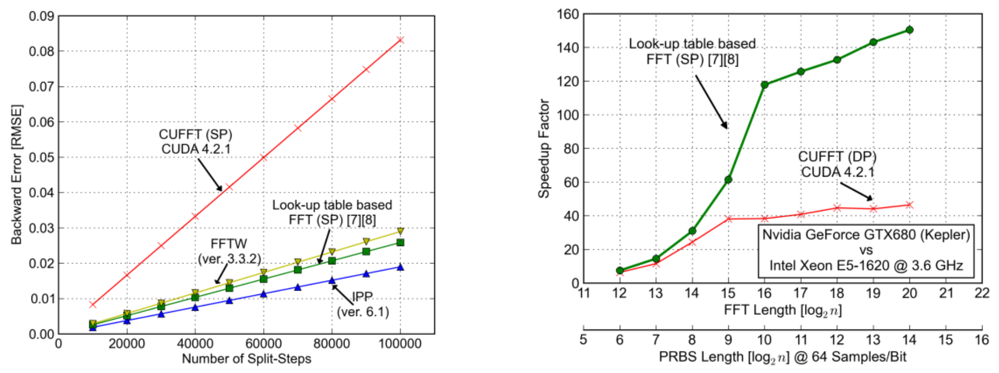

Fig. 2. Left: Accumulation of the backward error vs. the number of split-steps for an FFT-length of 2^{20} (using single precision). CPU implementations: FFTW and Intel IPP. GPU implementations: CUFFT and look-up table based; Right: Speedup of GPU vs. CPU

3. Fields of Application

GPU-parallelization allows modeling of fiber-optic transmission systems for the first time also in the nonlinear regime with a high channel count in a reasonable time. With the help of such GPU-based simulations an 80-channel transmission system consisting of low dispersion fibers (such as DSF or NZDSF) has been investigated recently, and it turned out that significant nonlinear interaction also occurs between channels with a high spectral distance [13]. Up to now only numerical simulations with a much lower channel count seemed feasible due to the prohibitively large computational times.

Another interesting field of application for GPU-parallelization is the investigation of the FEC performance (e.g. regarding error-floors) [14], which requires simulating a very high number of bits when using Monte-Carlo performance characterization. Also for the modeling of digital signal processing, which is commonly applied in e.g. coherent receivers, GPU-parallelization may be employed. A frequently used algorithm in that field is the least mean squares algorithm (LMS), e.g. for adapting the filter weights of an equalizer. Especially for large matrix sizes a parallelized solution can be preferable, and significant speedup factors compared to CPUs of more than 700 have been reported [15] on a GPU.

Finally it shall be mentioned that apart from parallelizing algorithms on a GPU also other interesting options exist to speed up simulations. In many of today’s PCs multi-core CPUs are installed. It is possible to automatically translate the kernels originally developed for GPUs into efficient code executed on multi-core CPUs [16] permitting parallelization within a CPU. An alternative could be to employ cloud computing services, which have become popular in recent years, and some providers already offer on-demand establishment of elastic compute services comprising general-purpose computation on GPUs [17].

4. Conclusion

Parallelization of system modeling on GPUs has been discussed. It has been shown that the highest speedup of more than a factor of 140 compared to CPUs (using a single processing core) can be obtained in single precision arithmetic. The associated inaccuracies can be reduced by pre-computing the trigonometric function values for the FFTs in double precision. Additionally, a stratified Monte-Carlo sampling technique can be employed for combining simulations with single and double precision arithmetic to meet a pre-defined accuracy level.

5. Acknowledgements

The author would like to thank Adam Chachaj of Krohne Messtechnik, Heinrich Müller and Peter Krummrich of TU Dortmund and Michael Eiselt of ADVA Optical Networking for fruitful discussions. Many simulations have been performed with the commercial fiber-optic system simulation tool *PHOTOSS* described in [18].

References

[1] S. Zoldi, V. Ruban, A. Zenchuk, S. Burtsev, “Parallel Implementation of the Split-step Fourier Method For Solving Nonlinear Schrödinger Systems”, Society for Industrial and Applied Mathematics (SIAM) news **32** (1999).

[2] G. E. Moore, “Cramming more components onto integrated circuits”, Electronics **38** (1965).

[3] D. Geer, “Taking the Graphics Processor beyond Graphics”, IEEE Computer **9**, 14-16 (2005).

[4] G. Agrawal, “Nonlinear Fiber Optics”, 3rd. ed., Academic Press (2001).

[5] N. K. Govindaraju, B. Lloyd, Y. Dotsenko, B. Smith, J. Manferdelli, “High Performance Discrete Fourier Transforms on Graphics Processors”, Proc. of IEEE conference on Supercomputing (SC), article no. 2 (2008).

[6] D. Kirk, W.-M. W. Hwu, “Programming Massively Parallel Processors: A Hands-On Approach”, Morgan Kaufman (2010).

[7] S. Pachnicke, “Fiber-Optic Transmission Networks: Efficient Design and Dynamic Operation”, Springer (2011).

[8] S. Pachnicke, A. Chachaj, C. Remmersmann, P. M. Krummrich, “Fast Parallelized Simulation of 112 Gb/s CP-QPSK Transmission Systems using Stratified Monte-Carlo Sampling”, Proc. of OFC, paper OWO2 (2011).

[9] J. C. Schatzman, “Accuracy of the Discrete Fourier Transform and the Fast Fourier Transform”, SIAM J. Scientific Comput. **17**, 1150-1166 (1996).

[10] M. Frigo, S. G. Johnson, “The design and implementation of FFTW3“, Proc. of the IEEE **93**, 216–231 (2005).

[11] S. Hellerbrand, N. Hanik, “Fast Implementation of the Split-Step Fourier Method using a Graphics Processing Unit”, Proc. of OFC, paper OTuD7 (2010).

[12] P. Serena, N. Rossi, M. Bertolini, A. Bononi, “Stratified Sampling Monte Carlo Algorithm for Efficient BER Estimation in Long-Haul Optical Transmission Systems”, IEEE J. Lightw. Technol. **27**, 2404-2411(2009).

[13] C. Xia, W. Schairer, A. Striegler, L. Rapp, M. Kuschnerov, J. F. Pina, D. van den Borne, “Impact of Channel Count and PMD on Polarization-Multiplexed QPSK Transmission”, IEEE J. Lightw. Technol. **29**, 3223-3229 (2011).

[14] G. Falcao, V. Silva, L. Sousa, “How GPUs can outperform ASICs for fast LDPC decoding”, Proc. of ACM International Conference on Supercomputing (ICS), 390-399 (2009).

[15] W. Bozejko, A. Dobrucki, M. Walczynski, “Parallelizing of digital signal processing with using GPU”, Proc. of IEEE Signal Processing Algorithms, Architectures, Arrangements, and Applications conference (SPA), pp. 29-33 (2010).

[16] J. A. Stratton, S. S. Stone, W.-M. W. Hwu, “MCUDA: An Efficient Implementation of CUDA Kernels for Multi-core CPUs”, Lecture Notes in Computer Science **5335**, 16-30 (2008).

[17] R. R. Exposito, G. L. Taboada, S. Ramos, J. Tourino, R. Doallo, “General-purpose computation on GPUs for high performance cloud computing”, Wiley J. Concurrency and Computation **24** (2012).

[18] M. Windmann, S. Pachnicke, E. Voges, “PHOTOSS: The simulation tool for optical transmission systems”, Proc. of SPIE **5247**, 51-60 (2003).