Fast Implementation of the Split-Step Fourier Method using a Graphics Processing Unit

Stephan Hellerbrand and Norbert Hanik

Technische Universität München, Institute for Communications Engineering, D-80290 Munich, Germany
stephan.hellerbrand@tum.de

Abstract: We describe how simulations based on the Split-Step Fourier Method can be significantly accelerated by using commodity graphics hardware. Results for a benchmark scenario are provided to highlight performance and accuracy.

OCIS codes: (060.4510) Optical communications, (060.2330) Fiber optics communications

1. Introduction

The simulation of optical transmission systems plays an important role both in research and link design. Solving the nonlinear Schrödinger equation, which describes the signal evolution in an optical fiber, is usually the most time-consuming task. This holds in particular for long test signals, which have to be used to account for the channel memory in high-speed transmission systems [1]. Therefore the optimization of optical communication links can take very long even if effective optimization algorithms are used [2]. In addition to that, in modern phase modulated and polarization multiplexed systems the nonlinear interaction of signal and noise as well as polarization mode dispersion (PMD) can often not be neglected. For systems in which semi-analytical models such as [3] are not applicable, Monte-Carlo sampling has to be used. The corresponding large number of evaluations of the signal propagation can lead to a very large time consumption. In this article we describe how the computation of the signal propagation in an optical fiber by the Split-Step Fourier Method (SSFM) [4] can be significantly accelerated using graphics processing units (GPU). We concentrate on commodity graphics hardware, which can be programmed via the NVIDIA Compute Unified Device Architecture (CUDA) [5].

2. Parallel Implementation of the Split-Step Fourier Method using a GPU

Over the past few years the increase of the raw computational power delivered by GPUs has exceeded that of central processing units (CPU) by far [6]. The task of graphics rendering has led to a highly parallel chip-architecture, which provides this performance at comparably low clock rates.

A significant amount of the operations performed in the SSFM lends itself to a parallel implementation. This has already been highlighted in [7]. In particular, the parallel implementation of the linear operator in the Fourier domain as well as the nonlinear parameter in the time domain is straightforward. The signal vector $\mathbf{u}$ can be split up into N_{proc} blocks containing N/N_{proc} elements. The element-wise multiplications can then be performed in parallel by N_{proc} processing units. In addition to that, the inherent parallelism in the FFT operation is exploited by optimized FFT routines, which are available as part of the CUDA framework [8].

In CUDA terminology, the CPU/RAM are referred to as *Host* and the GPU/VRAM as *Device*. The GPU based SSFM is implemented in the following way. At the beginning of the program, the signal vector is stored on the *Host*. In order to offload the computation to the GPU in a first step memory has to be allocated and the signal vector is then transferred onto the *Device*. Now the step size for the first symmetric step has to be found, which is done by the maximum nonlinear phase rotation ϕ_{max} [9]. After that the first nonlinear step over $\Delta z/2$ is computed, which involves computation of the nonlinear operator and a subsequent element wise multiplication with the signal vector. Then the signal is transformed into the frequency domain for the dispersive step over Δz by using [8]. After another element-wise multiplication the signal vector is transformed back into the time-domain, where the second nonlinear step over $\Delta z/2$ concludes the symmetric step. This procedure is then iterated and when the end of the fiber segment is reached the signal is copied back onto the *Host*.

3. Evaluation of Performance and Accuracy

In order to determine the achievable savings in computation time we have carried out simulations of an optical transmission system for a program using only the CPU and for a program which offloads the computation of the SSFM onto the GPU.

OTuD7.pdf

The test hardware was a workstation equipped with a Core 2 Quad processor Q9650 (3 GHz, 6MB cache, 4 cores) and 8 GB RAM. The graphics hardware was a GTX285 graphics card by ZOTAC, which features an NVIDIA GT200b chip with 240 single precision cores and 30 double precision cores. The additional cost for the graphics hardware was a small fraction of the cost for the workstation. The SSFM code was implemented in C in both cases and the GPU supported version also used CUDA [5]. The FFT on the CPU was implemented by the FFTW library [10] and on the GPU the CUFFT library was used. The other components of the simulation were implemented in MATLAB. Both implementations of the SSFM were called from MATLAB via the mex-interface.

We have simulated a 10.7 Gb/s RZ-DPSK system for comparison of the functions, in which RZ pulses according to a de Bruijn pseudorandom binary sequence (DBBS) are transmitted. The link comprises 80km spans of SSMF (D=16ps/nm/km, γ=1.3W^{-1}km^{-1}, α=0.2dB/km), the dispersion of which is fully compensated by using DCF (D=-80ps/nm/km, γ=4.8W^{-1}km^{-1}, α=0.5dB/km, L=16km). After each span an ideal amplifier fully compensates the loss the signal has experienced. The temporal resolution is 32 points per symbol.

The first series of simulations was performed to evaluate the potential time saving to be achieved by using the GPU. The simulation of transmission over N_{span}=30 spans was carried out for increasing order of the DBBS both for the program using only the CPU and for the GPU assisted code. Two version of the GPU based function were tested: one version uses single precision computations and the other one uses double precision. After the simulation the time consumed only by the SSFM program was measured and the relative speed-up is shown on the left side of Fig. 1.

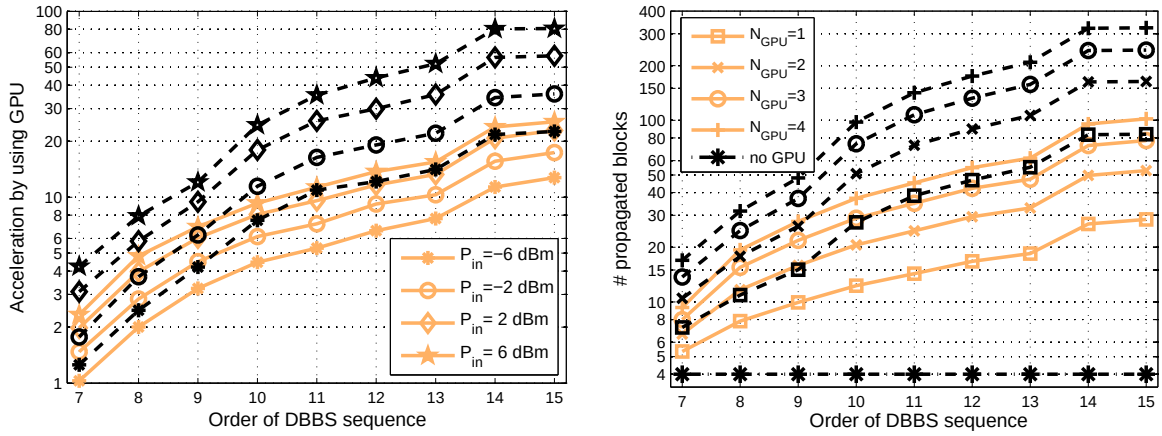

Fig. 1. Left: Acceleration of simulation versus order of the test sequence for transmission over $N_{\text{span}} = 30$ spans and $\phi_{\text{max}} = 5 \cdot 10^{-3}$. Right: Number of blocks finished in unit time for $P_{\text{in}} = 6$dBm. (Dashed lines indicate single-precision and solid lines indicate double-precision)

The first observation to be made is that a significant speed-up can be achieved. Longer signal vectors clearly result in a larger advantage for the GPU assisted program since more vector elements lead to a better utilization of the GPU. Higher input power P_{in} on average implies a larger number of steps. Therefore the number of the operations carried out on the graphics card in parallel more and more outweighs the computational overhead incurred on every span. Therefore larger power levels giver higher speed-ups. Due to the larger number of single precision cores the single precision GPU assisted program provides much larger speed up factors. The highest recorded speed-up factors were 80 for single precision and 25 for double precision for an order 15 sequence and a span input power of $P_{\text{in}} = 6$dBm.

This comparison was based on a single threaded FFT implementation on the CPU. A multi-threaded implementation (4 threads) for the CPU has only given a speed-up of the FFT by a factor < 1.5. If a series of simulations for different parameter sets are to be performed then the 4 cores of the CPU can achieve a speed up factor close to 4 by assigning one core to the simulation of one block. This type of parallelization is useful e.g. for grid search parameter optimization or the simulation of nonlinear signal-noise interaction. The speed-up for this kind of simulation is shown on the right side of Fig. 1 in terms of the number of configurations, which can be simulated in the time it takes one CPU core to compute a single configuration. In a conservative estimate we assume that one core is fully occupied managing the GPU based simulation and the remaining CPU cores can run additional simulations.

The objective of the second series of simulations was to determine the computational accuracy of the GPU based approach, in particular of the single precision SSFM. The root mean squared error (RMSE) and the difference in required OSNR (ROSNR) at BER $= 10^{-9}$ as a measure of accuracy. Again the simulation of transmission over 30 spans was carried out both using the GPU assisted code and the program, which uses only the CPU. The length of

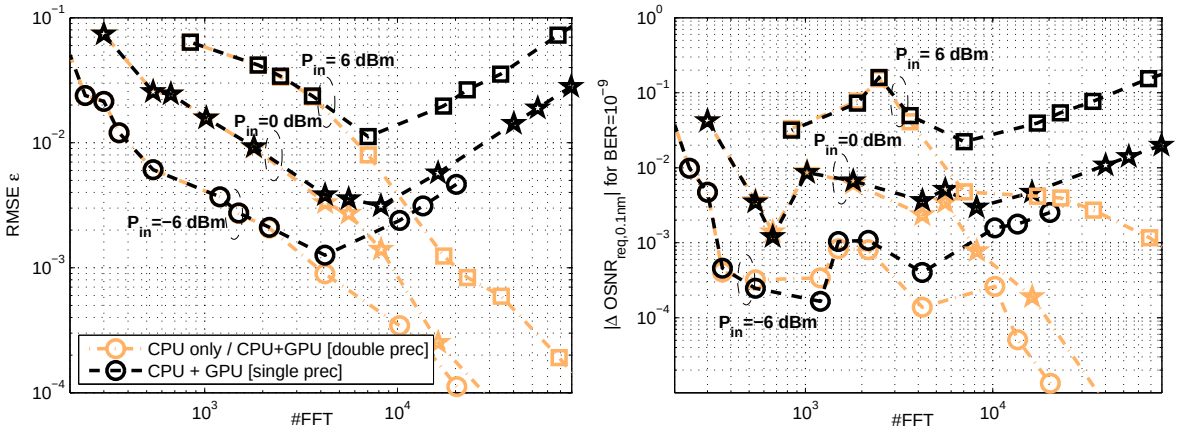

Fig. 2. Accuracy for the simulation of the 10.7 Gb/s RZ-DPSK system over 30 spans measured in terms RMSE (left) and the deviation from the $\mathrm{ROSNR}_{0.1\mathrm{nm}}$ for a Bit-Error-Rate of 10^{-9} in dB (right). Reference waveform was computed on the CPU with $\phi_{\max} = 10^{-6}$.

the DBBS was 64 and $\phi_{\max} \in [0.0008, \ldots, 0.05]$ was varied to tune the accuracy of the SSFM. The result in terms of RMSE is shown on the left side of Fig. 2.

For the program using only the CPU the accuracy improves continuously with increasing N_{FFT}, i.e. decreasing $\phi_{\max}$ and hence decreasing step-size Δz [9]. The accuracy for the double precision GPU assisted program completely overlaps with the CPU only program for the range of investigated $\phi_{\max}$. The results for the single precision GPU assisted program, however, become worse beyond a certain step-size. This is due to the accumulated error incurred by the single precision FFT operation.

On the right side of Fig. 2 the corresponding results in terms of ROSNR difference in dB are shown. A correlation with the results in RMSE can be observed. The fluctuations for very small ΔROSNR can be attributed to the accuracy of the ROSNR estimation. However, the difference is found to be smaller than 0.2dB in all investigated scenarios. Hence, we believe that single precision will be sufficient for many scenarios. Of course a suitable strategy has to be applied to identify these scenarios.

4. Conclusion

We have shown that the SSFM can be significantly accelerated by the use of commodity graphics hardware. The implementation of the SSFM on an off-the-shelf GPU by NVIDIA was compared to an implementation, which relied upon a quad-core CPU. We have found substantial acceleration for the single and the double precision implementation on the GPU compared to the CPU based program.

Simulations revealed that there is practically no difference between the results of the double precision GPU program and the simulation using only the CPU. Single precision computation is less accurate but gives sufficiently accurate results in many scenarios.

References

[1] L. Wickham, R.-J. Essiambre, A. Gnauck, P. Winzer, and A. Chraplyvy, "Bit pattern length dependence of intrachannel nonlinearities in pseudolinear transmission," *Photonics Technology Letters, IEEE*, vol. 16, no. 6, pp. 1591–1593, June 2004.

[2] L. D. Coelho, O. Gaete, and N. Hanik, "An algorithm for global optimization of optical communication systems," *AEU - International Journal of Electronics and Communications*, vol. 63, no. 7, pp. 541 – 550, 2009.

[3] L. D. Coelho, L. Molle, D. Gross, N. Hanik, R. Freund, C. Caspar, E. Schmidt, and B. Spinnler, "Modeling Nonlinear Phase Noise in Differentially Phase-Modulated Optical Communication Systems," *Optics Express*, vol. 17, no. 5, pp. 3226–3241, March 2009.

[4] G. P. Agrawal, *Nonlinear Fiber Optics*, 4th ed. Academic Press, 2007.

[5] *NVIDIA CUDA - Programming Guide*, NVIDIA Corp., 2009. [Online]. Available: http://www.nvidia.com/cuda

[6] D. Geer, "Taking the Graphics Processor beyond Graphics," *Journal of the IEEE Computer Society*, vol. 9, no. 9, pp. 14–16, September 2005.

[7] S. Zoldi, V. Ruban, A. Zenchuk, and S. Burtsev, "Parallel Implementation of the Split-step Fourier Method for Solving Nonlinear Schrödinger Systems," *SIAM News*, pp. 1–5, 1999.

[8] *CUDA - CUFFT Library, Version 2.1*, NVIDIA Corp., April 2008. [Online]. Available: http://www.nvidia.com/cuda

[9] O. V. Sinkin, R. Holzlöhner, J. Zweck, and C. Menyuk, "Optimization of the Split-Step Fourier Method in Modeling Optical-Fiber Communications Systems," *IEEE Journal of Lightwave Technology*, no. 1, pp. 61–68, January 2003.

[10] M. Frigo and S. G. Johnson. (2008) Fastest Fourier Transform in the West (FFTW). Webpage. [Online]. Available: http://www.fftw.org/