Fast Parallel Simulation of Fiber Optical Communication Systems Accelerated by a Graphics Processing Unit

S. Pachnicke[1], *Member*, *IEEE*, A. Chachaj[1,2], M. Helf[1], and P. M. Krummrich[1], *Member*, *IEEE*
[1]TU Dortmund, High Frequency Institute, Friedrich-Wöhler-Weg 4, 44221 Dortmund, Germany
Tel: +49 (231) 755 6675, Fax: +49 (231) 755 4631, e-mail: stephan.pachnicke@tu-dortmund.de
[2]TU Dortmund, Chair for Computer Graphics, Otto-Hahn-Str. 16, 44221 Dortmund, Germany

ABSTRACT
A parallel implementation of the split-step Fourier method utilizing the general purpose parallel computing architecture for graphics processing units CUDA is presented. Results of the GPU-implementation are compared to a conventional CPU-based approach regarding computation time and accuracy. We developed a novel implementation with a significantly higher accuracy than the CUDA intrinsic FFT in single precision mode yielding a high speed-up factor of up to 144 compared to a CPU implementation.
Keywords: Optical fiber communication, Optical propagation in nonlinear media, Parallel algorithms, Partial differential equations, Simulation.

1. INTRODUCTION

Modelling and simulation play a major role in today's research and development of optical networks. Numerical simulations of optical fiber links are typically based on the split-step Fourier method (SSFM), which unfortunately requires a high computational effort. Especially for transmission systems with significant nonlinear effects in the fiber, a high number of wavelength division multiplex (WDM) channels and a large transmission distance the computation time for propagation of the signal through the link falls in the range of several hours or days on a state of the art desktop computer. Such a PC is the standard equipment for a system designer. This is why in the past several attempts have been made to speed up SSFM-based simulations e.g. by introducing IIR filter structures that approximate the linear operator of the nonlinear Schrödinger equation [1] or by a split-step wavelet collocation method [2].

Most commercial simulation tools, however, are still based on the standard SSFM because it features the highest reliability. Another possibility to speed up the simulation time is to use a parallel implementation of the standard SSFM [3] and perform the simulation on multiple cores. Current quad-core CPUs (e.g. Intel® Core i7 965 XE) achieve a peak performance of more than 70 GFlop/s and graphics cards (e.g. NVIDIA GeForce GTX 275 with 240 stream processors) already provide more than 1 TFlop/s. This is why parallel processing on graphics processing units (GPU) is becoming an interesting alternative to accelerate numerical simulations. For this purpose NVIDIA provides a software platform called Compute Unified Device Architecture (CUDA) [4] to facilitate parallel processing on GPUs. CUDA already provides fast Fourier transform (FFT) functions in a GPU-optimized library. We implemented the entire SSFM on a 64-bit NVIDIA GeForce GTX 275 graphics card with 896 MB GDDR3 RAM (approximately 95 GFlop/s using double precision arithmetic on 30 stream processors and 1 TFlops/s using single precision arithmetic on 240 stream processors) and report on the accuracy of the results and the speedup compared to a standard single CPU-based SSFM approach (using the highly optimized C++ library Intel® IPP FFT [5] on an Intel® Core2 Quad CPU Q6600 (approximately 5.25 GFlop/s per core) with 2.4 GHz clock rate and 2GB of DDR2-800 RAM) for several reference scenarios. For the first time to our knowledge we show a novel implementation of the FFT/IFFT in single precision mode yielding a significantly higher accuracy than the CUDA intrinsic FFT implementation (CUFFT) making this very fast implementation applicable to most fiber optical transmission system simulation scenarios. We observed a speedup of up to 144 in single precision mode with low loss in accuracy and a speedup of almost a factor of 50 in double precision mode with accuracy comparable to CPU-based implementations.

2. PARALLEL CUDA IMPLEMENTATION AND SIMULATION SETUP

The signal propagation in optical fibers can be described by the nonlinear Schrödinger equation (NLSE), which has the (scalar) form [6]

$$\frac{\partial A}{\partial z} + \underbrace{\left(\frac{j\beta_2}{2}\frac{\partial^2}{\partial t^2} - \frac{\beta_3}{6}\frac{\partial^3}{\partial t^3} \right)}_{\hat{L}} A = \underbrace{\left(-\frac{\alpha}{2} + j\gamma \left|A\right|^2 \right)}_{\hat{N}} A. \qquad (1)$$

In equation (1) A is the complex envelope of the electric field, β_2 the group-velocity dispersion coefficient, β_3 the third-order dispersion coefficient, α the attenuation constant and γ the nonlinearity constant. The split-step approach divides the NLSE into a linear operator $\hat{L}$ and a nonlinear operator $\hat{N}$. Both operators are applied

independently. This is a good approximation, if the discretization steps Δz are chosen sufficiently small. As a simplified notation, the differential equation can be written as:

$$\frac{\partial A}{\partial z} = \left(-\hat{L} + \hat{N}\right) A \tag{2}$$

where the attenuation can either be computed in the linear or the nonlinear part. The linear operator $\hat{L}$ is more conveniently applied to the signal in the frequency domain, whereas the nonlinear operator $\hat{N}$ is applied to A in the time domain. The transformation between time and frequency domains and vice versa is obtained by FFTs and IFFTs, respectively. In our simulations for these FFTs/IFFTs we use a parallel CUDA implementation. We implemented the complete SSFM on the GPU so no data transfer between the graphics card and the external memory is necessary during the simulation of an entire fiber span. For the simulations we used CUDA runtime version 2.3 (driver version 196.21).

For the efficiency of the SSFM, the right choice of the step-size is crucial. If the step-size is chosen too large, the results will be inaccurate due to insufficient consideration of the interaction of the linear and nonlinear operators. On the other hand, if the step-size is chosen too small, the computational efficiency is decreased. If the error has to remain small, the nonlinear phase shift must not exceed a certain level, e.g. we used 8 mrad in our simulations. In detailed analyses on the SSFM this number has been found to yield a small global relative error [7]. In addition the step-size has been limited by a maximum of allowed artificial four-wave mixing (FWM) [8]. We used a conservative value of -50 dB of maximum allowed artificial FWM. In our simulations the smaller upper limit of both criteria has been used to bound the step-size.

For the assessment of the accuracy of the simulations we selected the following reference scenario (Fig. 1).

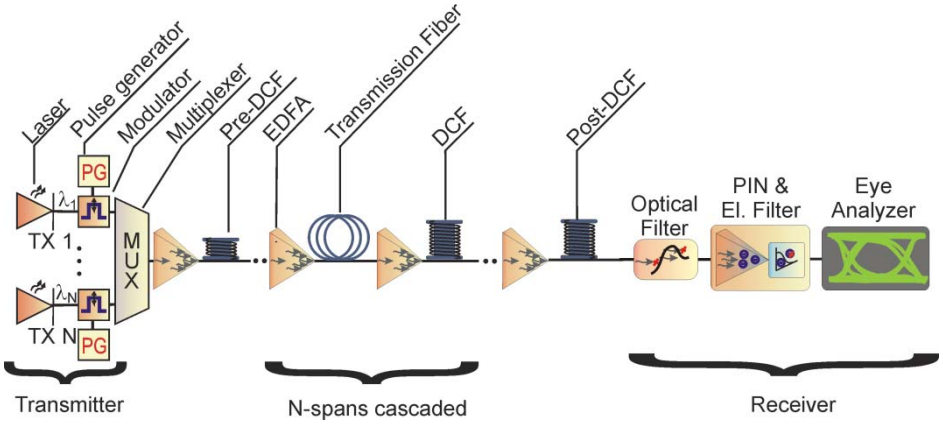

Figure. 1. Setup of the simulated transmission system.

The transmission system operated at a line rate of 10.7 Gb/s (NRZ-OOK) on a 50 GHz grid as an example for long-haul WDM transmission with considerable impact of nonlinear interchannel effects. We deployed a pre-DCF with 650 ps/nm. A distributed undercompensation scheme has been used along the transmission line with 50 ps/nm residual dispersion per span. At the receiver the accumulated dispersion has been driven to zero. SSMF has been assumed as transmission fiber. The launch power has been varied between -6 and 4 dBm/channel. The EDFAs in our simulations had a noise figure of 5.5 dB. We used a 50 GHz 2nd order Gaussian optical bandpass filter to select the desired channel and a 7 GHz 10th order electrical Bessel filter. 10 WDM channels and 25 fiber spans of 80 km length each have been simulated. In all simulations 64 samples/bit have been used.

As a figure of merit we calculated the Q-factor of the center channel for the different setups and the two different simulation methods (GPU- and CPU-based). An analytical noise representation [9] has been used separating numerical simulation of the signal and analytical representation of the noise variance. At the receiver a Q-factor is derived assuming Gaussian distribution of the noise. Other noise sources – apart from ASE – such as lasers or photodiodes have been ignored.

3. RESULTS

In our studies we investigated setups with different numbers of spans and different FFT sizes (respectively PRBS lengths) starting with a double precision (DP) implementation based on the CUDA intrinsic CUFFT library. We defined the Q-factor deviation ΔQ as the absolute value of the difference between CPU-based simulations and GPU-based simulations using the same random generator seed thus theoretically leading to identical results on the same hardware architecture.

$$\Delta Q \ [\mathrm{dB}] = \left| 20 \cdot \log_{10}\left(Q_{CPU,\,Double\,\mathrm{Prec}}\right) - 20 \cdot \log_{10}\left(Q_{GPU}\right) \right| \tag{3}$$

The error has been investigated at different span counts for a fixed launch power of 0 dBm/ch (Fig. 2, left) and

for different launch powers at a given transmission distance of 20 spans (Fig. 2, right). As a reference the simulated Q-factor has been approximately 11.3 dB for a launch power of 1 dBm/ch and a transmission distance of 20 spans. As can be seen from Fig. 2 (left and right) the Q-factor difference between the CPU- and GPU-based simulations is negligible for all investigated FFT-lengths ($\Delta Q < 10^{-9}$ dB).

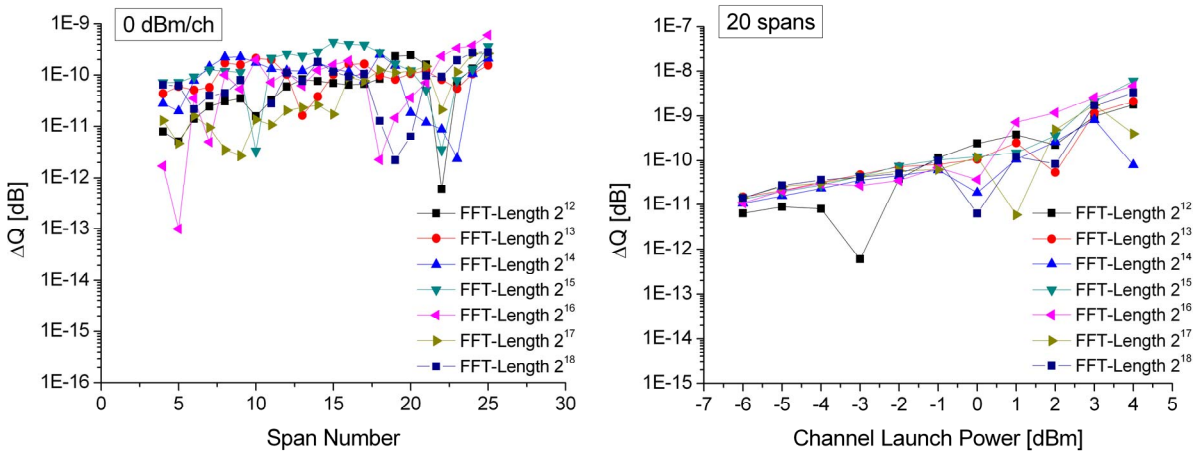

Figure 2. Q-factor deviation between CPU and GPU-based simulations for different FFT lengths vs. the transmission distance in spans (left) and vs. the channel launch power at a given transmission distance (right). Both figures show results obtained in double precision mode.

The depicted Q-factor deviations in Fig. 2 (left) start at a span count of 4 because for a lower number of transmission spans the signal quality is almost ideal, and no Q-factor could be determined (error free transmission). Figure 2 (left) shows that higher FFT lengths do not necessarily lead to a (significantly) higher Q-factor deviation. However, it can be observed that the error increases with the span count, which is due to the increasing number of split-steps (or FFT/IFFT operations). To prove that the observed Q-factor deviation stems from the FFT/IFFT operation, we used a mixed GPU/CPU implementation computing the FFT/IFFTs on the CPU and the other operations on the GPU showing that the error vanishes for that configuration.

Another analysis is shown in Fig. 2 (right). Here the transmission distance has been fixed, and the channel launch power in each transmission fiber has been varied. It can be observed that the average Q-factor deviation increases from approximately 10^{-11} dB for a launch power of -6 dBm/ch to approximately 10^{-9} dB for a launch power of +4 dBm/ch. This is in line with the increase in the number of split-steps (504 per span for -6 dBm/ch to 50295 per span for +4 dBm/ch) due to the maximum artificial FWM criterion. Again no significant dependence on the FFT length could be observed.

Figure 3 (left) shows the speedup factor for GPU-based simulations compared to CPU-based simulations. It can be observed from this figure that for all investigated scenarios the simulation time of the GPU-based implementation is lower than the CPU-based implementation (the speedup factor is always larger than 1). Furthermore, the speedup is increasing with a longer FFT-size. The reason for this is that a higher FFT length allows a better parallelization on the GPU architecture. We did not observe a significant (speedup) difference between different launch powers. For an FFT-size of 2^{18} a speedup of almost a factor of 50 has been measured with double precision accuracy. The simulation time one the CPU for such an FFT-size and a launch power of +4dBm/ch is already 116 min for a single fiber span compared to 139 s with the GPU-based implementation.

In a second set of simulations we investigated simulations based on single precision (SP) arithmetic. SP allows an additional speedup (compared to DP) on current GPUs due to the fact that 8 SP processors are available for 1 DP unit. However, the CUDA intrinsic FFT (CUFFT), exhibits a significantly higher error than CPU-based implementations (both implementations are compared in SP). To test, whether better accuracies can be achieved even for SP we developed our own FFT based on an algorithm shown in [10]. To improve the accuracy of the results we conducted several tests showing that the main reason for the higher error in SP mode is due to the use of an intrinsic implementation of trigonometric functions on the GPU not compliant to the IEEE standard. To avoid this problem we use an alternative implementation of the FFT where the twiddle factors (which are based on trigonometric functions) are precomputed on the CPU with high accuracy and stored in the graphics memory leading to a slightly lower performance for high FFT lengths than CUFFT as shown in Fig. 3 (left).

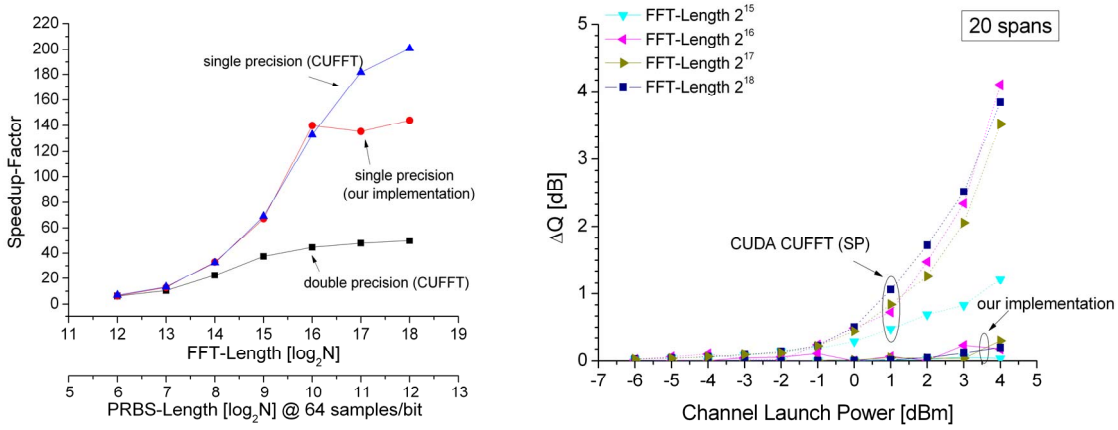

Figure 3. Speedup factor for GPU-based compared to CPU-based simulations in double precision on a single core (left) and Q-factor deviation between CPU and GPU-based simulations in single precision for different FFT lengths vs. channel launch power (right).

Results for SP simulations are depicted in Fig. 3 (right). Only FFT-lengths of 2^{15} to 2^{18} are shown because SP outperforms DP only for higher FFT-lengths. The reason is that for shorter FFT lengths the fiber simulation time is dominated by copying all data from external memory to graphics memory. Our simulations show that the Q-factor deviation of our implementation remains below a value of 0.25 dB (equivalent to 2.9% in non-logarithmic units) for the investigated scenarios. However, the error is significantly higher than the one shown in Fig. 2 (right) due to the use of SP. Compared to the intrinsic CUFFT implementation in single precision, though, our algorithm shows a much higher accuracy with the Q-factor deviation in the CUFFT implementation growing exponentially making it unusable for many simulation scenarios (especially with higher launch powers of more than 0 dBm/ch). If a Q-factor deviation of up to 0.25 dB can be tolerated in the simulations, our implementation provides a fast alternative with speedup of up to a factor of 144 compared to CPU-based simulations.

4. CONCLUSIONS

We have studied GPU-based parallel implementations of the split-step Fourier method to accelerate numerical simulations of fiber optical transmission systems. Our GPU-enhanced simulations allow a speedup of up to a factor of 50 in double precision with negligible error compared to standard single CPU-based simulations and an even higher speedup of up to 144 in single precision with maximum observed error of 2.9% for typical simulation scenarios with an FFT-size of 2^{18}.

REFERENCES

[1] M. Plura *et al.*: Improved split-step method for efficient fibre simulations, *IEE Electron. Lett.*, vol. 37, no. 5, pp. 286-287, Mar. 2001.

[2] T. Kremp *et al.*: Fast split-step wavelet collocation method for WDM system parameter optimization, *IEEE/OSA J. Lightw. Technol.*, vol. 23, no. 3, pp. 1491-1502, Mar. 2005.

[3] S. Hellerbrand *et al.*: Fast implementation of the split-step Fourier method using a graphics processing unit, *Proc. of OFC/NFOEC,* OTuD7, Mar. 2010.

[4] M. Garland *et al.*: Parallel computing experiences with CUDA, *IEEE Micro*, vol. 28, no. 4, pp. 13- 27, Jul. 2008.

[5] R. Inkol *et al.*: Applications of performance benchmarking to the development of signal processing systems based on personal computer technology, *IEEE Canadian Conference on Electrical and Computer Engineering*, pp. 41-45, May 2006.

[6] G. P. Agrawal*: Nonlinear Fiber Optics*, 3rd ed., Academic Press, San Diego, 2001.

[7] O. V. Sinkin *et al.*: Optimization of the split-step Fourier method in modeling optical-fiber communications systems, *IEEE/OSA J. Lightw. Technol.*, vol. 21, no. 1, pp. 61-68, Jan. 2003.

[8] G. Bosco *et al.*: Suppression of spurious tones in fiber systems simulations based on the split-step method, *IEEE LEOS Annual Meeting*, WH 4, 1999.

[9] M. Windmann *et al.*: PHOTOSS: The simulation tool for optical transmission systems, *Proc. SPIE*, vol. 5247, pp. 51-60, Sep. 2003.

[10] N. K. Govindaraju *et al.*: High performance discrete Fourier transforms on graphics processors, *Proc. of ACM/IEEE Conference on Supercomputing*, pp. 1-12, Nov. 2008.