Dissertation

submitted to the
Combined Faculty for the Natural Sciences and Mathematics
of Heidelberg University, Germany
for the degree of
Doctor of Natural Sciences

Evaluation of single photon avalanche diode arrays for imaging fluorescence correlation spectroscopy.

FPGA-based data readout and fast correlation analysis on CPUs, GPUs and FPGAs.

Submitted by
Dipl.-Phys. Jan Buchholz
born in Bad Kreuznach, Germany

Dissertation

submitted to the
Combined Faculty for the Natural Sciences and Mathematics
of Heidelberg University, Germany
for the degree of
Doctor of Natural Sciences

Submitted by
Dipl.-Phys. Jan Buchholz
born in Bad Kreuznach, Germany

Date of the oral examination: .....

Evaluation of single photon avalanche diode arrays for imaging fluorescence correlation spectroscopy.

FPGA-based data readout and fast correlation analysis
on CPUs, GPUs and FPGAs.

Advisors: Prof. Dr. Udo Kebschull
Prof. Dr. Jörg Langowski

Es wird ja fleißig gearbeitet und viel mikroskopiert,
aber es müßte mal wieder einer einen gescheiten Gedanken haben.
RUDOLF VIRCHOW

Abstract

The metabolism of all living organisms, and specifically also of their smallest constituents, the cell, is based on chemical reactions. A key factor determining the speed of these processes is transport of reactants, energy, and information within the and between the cells of an organism. It has been shown that the relevant transport processes also depend on the spatial organization of the cells. Such transport processes are typically investigated using fluorescence correlation spectroscopy (FCS) in combination with fluorescent labeling of the molecules of interest. In FCS, one observes the fluctuating fluorescence signal from a femtoliter-sized sub-volume within the sample (*e.g.*, a cell). The variations in the intensity arise from the particles moving in and out of this sub-volume. By means of an autocorrelation analysis of the intensity signal, conclusion can be drawn regarding the concentration and the mobility parameters, such as the diffusion coefficient. Typically, one uses the laser focus of a confocal microscope for FCS measurements. But with this microscopy technique, FCS is limited to a single spot a every time. In order to conduct parallel multi-spot measurements, *i.e.*, to create diffusion maps, FCS can be combined with the lightsheet based selective plane illumination microscopy (SPIM). This recent widefield microscopy technique allows observing a small plane of a sample ($1-3\,\mu\text{m}$ thick), which can be positioned arbitrarily. Usually, FCS on a SPIM is done using fast electron-multiplying charge-coupled device (EMCCD) cameras, which offer a limited temporal resolution ($\sim 500\,\mu\text{s}$). Such a temporal resolution only allows measuring the motion of intermediately sized particles within a cell reliably. The limited temporal resolution renders the detection of even smaller molecules impossible. In this thesis, arrays of single photon avalanche diodes (SPADs) were used as detectors. Although SPAD-based image sensors still lack in sensitivity, they provide a significantly better temporal resolution ($1-10\,\mu\text{s}$ for full frames) that is not achievable with sensitive cameras and seem to be the ideal sensors for SPIM-FCS.

In the course of this work, two recent SPAD arrays (developed in the groups of Prof. Edoardo Charbon, TU Delft, the Netherlands, and EPFL, Switzerland) were extensively characterized with regards to their suitability for SPIM-FCS. The evaluated SPAD arrays comprise 32×32 and 512×128 pixels and allow for frame rates of up to 300 000 or 150 000 frames per second, respectively. With these specifications, the latter array is one of the largest and fastest sensors that is currently available. During full-frame readout, it delivers a data rate of up to $1.2\,\text{GiB}/\text{s}$. For both arrays, suitable readout-hardware-based on field programmable gate arrays (FPGAs) was designed. To cope with the high data rate and to allow real-time correlation analysis, correlation algorithms were implemented and characterized on the three major high performance computing platforms, namely FPGAs, CPUs, and graphics processing units (GPUs). Of all three platforms, the GPU performed best in terms of correlation analysis, and a speed of 2.6 over real time was achieved for the larger SPAD array.

Beside the lack in sensitivity, which could be accounted for by microlenses, a major drawback of the evaluated SPAD arrays was their afterpulsing. It appeared that the temporal structure superimposed the signal of the diffusion. Thus, extracting diffusion properties from the autocorrelation analysis only proved impossible. By additionally performing a spatial cross-correlation analysis such influences could be significantly minimized. Furthermore, this approach allowed for the determination of absolute diffusion coefficients without prior calibration. With that, spatially resolved measurements of fluorescent proteins in living cells could be conducted successfully.

Zusammenfassung

Für alle Organismen ist der Transport von Stoffen, Energie und Information Voraussetzung dafür, ihren komplexen Stoffwechsel, basierend auf chemischen Prozessen, aufrechtzuerhalten. Dies gilt insbesondere für die kleinsten lebenden Einheiten, die Zellen. Ein wesentlicher Faktor, der die Geschwindigkeit diese Prozesse bestimmt, ist der Transport von Reaktionspartner, Energie und Information innerhalb von und zwischen den Zellen eines Organismus. Es wurde bereits gezeigt, dass die relevanten Transportprozesse auch von der räumlichen Organisation der Zelle abhängen. Solche Transportprozesse werden typischerweise mit Hilfe von Fluoreszenz-Korrelations-Spektroskopie (FCS) in Kombination mit fluoreszenzmarkierten Molekülen untersucht. Hierbei beobachtet man die Fluktuationen des Fluoreszenzsignals in einem kleinen Femtoliter-großen Teilvolumen innerhalb der Probe (z.B.: eine Zelle). Die Intensitätsschwankungen des Signals kommen dadurch zustande, dass Moleküle in das Volumen hinein- und wieder herausdiffundieren. Mittels Autokorrelationsanalyse des Intensitätssignals lassen sich Rückschlüsse auf die Konzentration und Mobilitätseigenschaften, insbesondere den Diffusionskoeffizienten, ziehen. Typischerweise werden diese kleinen Volumina ($\sim 1\,\mu m^3$) durch einen Laserfokus erzeugt und mit einem konfokalen Laser-Mikroskop beobachtet. Der konfokale Aufbau lässt jedoch zu jedem Zeitpunkt Messungen nur an einem Ort zu. Um parallele ortsaufgelöste Messungen durchführen zu können, um beispielsweise Diffusionskarten zu erstellen, kann FCS etwa mit Lichtscheibenfluoreszenzmikroskopie (SPIM) kombiniert werden. Diese relative neue Weitfeldmikroskopietechnik erlaubt es eine beliebig positionierbare Ebene ($1-3\,\mu m$ Dicke) in einer Zelle zu beobachten. Typischerweise kommen dabei schnelle elektronenvervielfachende CCD-Kameras (EMCCD) zum Einsatz. Diese bieten nach aktuellem Stand den besten Kompromiss zwischen Sensitivität und maximaler zeitlicher Auflösung ($\sim 500\,\mu s$). Die begrenzte zeitliche Auflösung erlaubt allerdings nur das Auflösen der Bewegung von mittelgroßen Molekülen in einer Zelle. Sehr kleine Teilchen können damit nicht beobachtet werden. Im Rahmen dieser Arbeit wurden Arrays aus Einzelphoton-Lawinenphotodioden (SPADs) als Detektoren verwendet. Diese bieten eine für Kameras unerreichbare hohe zeitliche Auflösung ($1-10\,\mu s$ für ganze Bilder) und sind damit scheinbar ideale Detektoren für SPIM-FCS. Allerdings liegt der Nachteil bei der deutlich reduzierten Photosensitivität.

In der vorliegenden Arbeit wurden zwei dieser neuartigen SPAD-Arrays (entwickelt in der Gruppe von Prof. Edoardo Charbon, TU Delft, Niederlande, und EPFL, Schweiz) bezüglich der Eignung in einem SPIM-FCS ausführlich charakterisiert. Die verwendeten SPAD-Arrays umfassen 32×32 und 512×128 Pixel und können mit bis zu 300000 bzw. 150000 Bildern pro Sekunde ausgelesen werden. Das letztgenannte Array ist damit einer der größten und schnellsten Sensoren, wobei die Datenrate für Vollbilder im Bereich von $1{,}2\,GiB/s$ liegt. Die Ansteuerung und Auslese wurde in beiden Fällen mit Hilfe von rekonfigurierbarer Logik (FPGAs) realisiert. Um bei den hohen Datenraten der verwendeten Sensoren die FCS-Analyse in Echtzeit ausführen zu können, wurden die Korrelationsalgorithmen auf den drei wichtigsten Plattformen für Hochleistungsrechnen, nämlich FPGAs, CPUs und Grafikkarten (GPUs), implementiert und charakterisiert. Von allen drei Plattformen war die GPU hinsichtlich Korrelationsanalyse besonders performant und es wurde eine Geschwindigkeit von 2,6 oberhalb von Echtzeit für den größeren SPAD-array erreicht.

Neben der geringen Empfindlichkeit der untersuchten SPAD-Arrays, welche durch Mikrolinsen verbessert werden können, stellte sich das Nachpulsen (afterpulsing) als großer Nachteil der evaluierten SPAD-Arrays heraus. Dessen zeitliche Struktur überlagerte sich mit dem durch Diffusion verursachten Verhalten. Dies machte es unmöglich die Diffusionsprozesse aus der Autokorrelationsanalyse alleine zu extrahieren. Es zeigte sich jedoch, dass diese Einflüsse unter Zuhilfenahme der ramlichen Kreuzkorrelation deutlich minimiert werden konnten. Gleichzeitig war es damit möglich, absolute Diffusionskoeffizienten zu extrahieren. Schließlich konnten erfolgreich erste ortsaufgelöste Messungen von fluoreszierenden Proteinen in Zellen durchgeführt werden.

Table of Contents

1. Introduction

1.1. Imaging transport processes in live cells

1.1.1. Transport processes in living cells

The cell is the basic structural unit of all known living organisms. Typical animal cells, as sketched in Figure 1.1, are 15 µm in diameter, enclosing a volume of approximately 14 fL. Cells contain a dense aqueous solution, the cytoplasm, that is confined by a lipid bi-layer, the cell membrane. The cytoplasm contains various proteins, lipids, sugars and other small molecules [4].

All animal cells have membrane contained compartments that provide specialized reaction chambers (different pH) for many metabolic processes, like for example glycolysis. The concentration of the molecules varies from 10^3 to 10^9 copies per cell [4]. The genome, consisting of the deoxyribonucleic acid (DNA) molecules with a total length of approximately 2 m for humans, is located inside the nucleus of every single cell. The DNA is packed into the nucleus in a hierarchic manner.

To react, the partners have to be in contact. When chemical reactions are modeled, the underlying transport processes have to be taken into account. Three common transport processes occur in a living cell, as illustrated in Figure 1.2.

The simplest and one of the most important transport process of molecules is BROWNIAN motion (BM) (*cf.*, Figure 1.2a), where particles perform a random walk through a gel-like, or viscous, environment. Each time a particle collides with another, its speed and direction changes slightly. Figure 1.2a shows the typical quivering motion of a random walk [28, 60, 126, 170–172, 224].

A statistical description is needed to quantify the undirected, random motion. For a single particle moving along a trajectory $\vec{r}(t)$, the global behavior is well described by the mean squared displacement (MSD), which is defined as the time average over the squared displacements within the trajectory over different given time lags τ:

$$\mathrm{MSD}(\tau) \equiv \left\langle r^2(\tau) \right\rangle_t := \lim_{T\to\infty} \frac{1}{T} \int_0^T |\vec{r}(t+\tau) - \vec{r}(t)|^2 \, \mathrm{d}t. \tag{1.1}$$

From the stochastic single-particle view, a deterministic particle density description can be derived, which is known as FICK'S second law [60, 65]:

$$\frac{\partial c(\vec{r},t)}{\partial t} = D\nabla^2 c(\vec{r},t), \tag{1.2}$$

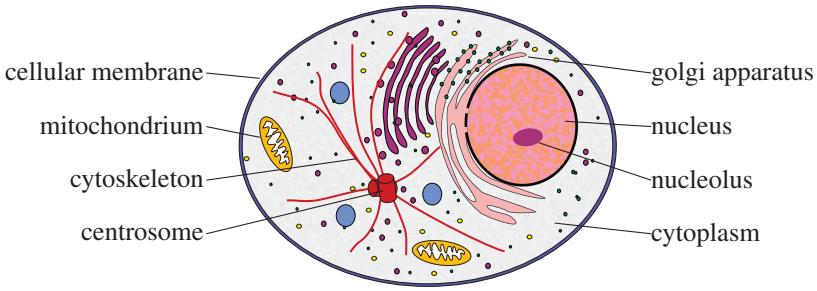


Figure 1.1.: Sketch of an animal cell. Drawing provided by JAN W. KRIEGER.

where D is the diffusion coefficient and $c(\vec{r},t)$ the concentration at position $\vec{r}$ and time t. A comparison of the MSD with the solution of FICK'S second law gives the following linear relation for particles undergoing BROWNIAN motion in a viscous medium in n spatial dimensions [60]:

$$\mathrm{MSD}(\tau) = 2n \cdot D \cdot \tau. \tag{1.3}$$

The diffusion coefficient is to random motion what the velocity is to linear movement: both define the distance a particle has traveled on average within a certain time.

For spherical particles with radius R_h diffusing through a liquid with viscosity η, temperature T and a low REYNOLDS number, the diffusion coefficient is given by the STOKES-EINSTEIN equation [60]:

$$D = \frac{k_B \cdot T}{6\pi \cdot \eta \cdot R_\mathrm{h}}, \tag{1.4}$$

where k_B is the BOLTZMANN constant. More precisely, R_h is the hydrodynamic radius of a particle which takes into account, that a sphere of any material will drag along some of its surrounding solvent molecules. Therefore it will appear to be slightly larger than its geometrical radius. Typical biological proteins have a hydrodynamic radius of $R_\mathrm{h} = 1\,\mathrm{nm} \ldots 10\,\mathrm{nm}$ [64], thus resulting in a diffusion coefficient of $D_{\mathrm{cell},\,20\,^\circ\mathrm{C}} \approx 60 \ldots 5\,\mu\mathrm{m}^2/\mathrm{s}$ (Equation (1.4)) assuming a viscosity of $\eta_\mathrm{cell} \approx 3 \cdot \eta_\mathrm{water}$ [21, 58, 71, 138].

In a dense environment like the cytoplasm, the free diffusion is hindered (Figure 1.2b). A large fraction of the cell is occupied by different molecules and large complexes (protein machinery, DNA, etc.). Within such a crowded space, the MSD is no longer a linear function of time but follows a power-law. This phenomenon is called anomalous diffusion [92, 148, 149, 232, 233]:

$$\mathrm{MSD}(\tau) = 2n \cdot \Gamma \cdot \tau^\alpha, \tag{1.5}$$

where Γ is the generalized anomalous diffusion coefficient and α is the anomaly parameter.

Finally, in some cases diffusion-based transport of molecules is not sufficient for the needs of a cell. The bigger the molecule, the slower its movement, and the longer the time the molecule needs to reach its destination. To maintain chemical reactions at sufficiently large rates and to direct particles to specific places, active transport is required (*e.g.*, motion of vesicles with neurotransmitters in a cm-long nerve-cell). This task is accomplished, for example, by motor proteins, that move along the internal structures of the cell (Figure 1.2c). These proteins gain velocities of up to $v = 100\,\mu\mathrm{m/s}$ (observed *e.g.*, in plants [195]). The MSD for active transport is described as:

$$\mathrm{MSD}(\tau) = (v \cdot \tau)^2. \tag{1.6}$$

Both, active transport and normal diffusion can be described as special cases of anomalous diffusion (either $\alpha = 1$ for BROWNIAN motion or $\alpha = 2$ for active transport).

(a) free BROWNIAN motion

(b) diffusion in a crowded environment

(c) directed motion

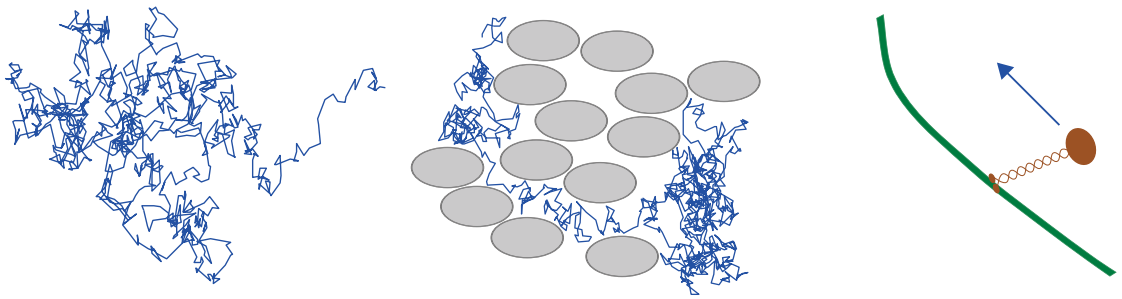


Figure 1.2.: Different transport processes inside a cell. (a) Free BROWNIAN motion. (b) BROWNIAN motion in a crowded environment. (c) Directed motion of a motor-protein along a cell structure.

1.1.2. Measuring transport processes

To measure the random motion of particles, they have to be made distinguishable from other particles in a cell. A common method to achieve this goal is to label them fluorescently. Such prepared molecules can then be observed by several methods.

Single particle tracking

The most straightforward approach to analyze the motion of a particle is by tracking its trajectory $\vec{r}(t)$ and to directly calculate the MSD. This method, called single particle tracking (SPT), is illustrated in Figure 1.3a. SPT measurements are either based on point-scanning confocal setups or on wide-field microscopy. In the latter case, a camera records a time series of images. The positions of the particles is registered in each frame and trajectories are obtained by merging particles over time. Although a pixel-based detector is used, the position can be determined with sub-pixel accuracy, that is better than the optical resolution. A two to ten times higher resolution is achieved by fitting a GAUSSIAN function to the image of each particle [106]. This method can be used to track molecules in a cell [51, 130, 207] or entire cells in a living organism [108].

Another recent technique makes use of a point-scanning confocal microscope [130]. The microscope detects the fluorescence from a small focal volume within the sample. Once a particle has been identified, a circular path around the particle is continuously scanned, as illustrated in Figure 1.3b. When the particle moves, the fluorescence intensity along the laser path is no longer constant but maximal in the direction of motion of the particle. The center of the scanning circle is then moved to the point of the highest intensity and thus tracks the particle's trajectory. Three-dimensional tracking is possible by extending the scanning along the z-axis.

Both techniques rely on very low sample concentrations to identify and follow single particles independently. Irrespective of the method used for SPT, these techniques lack in speed as many trajectories are necessary to gain reasonable statistics.

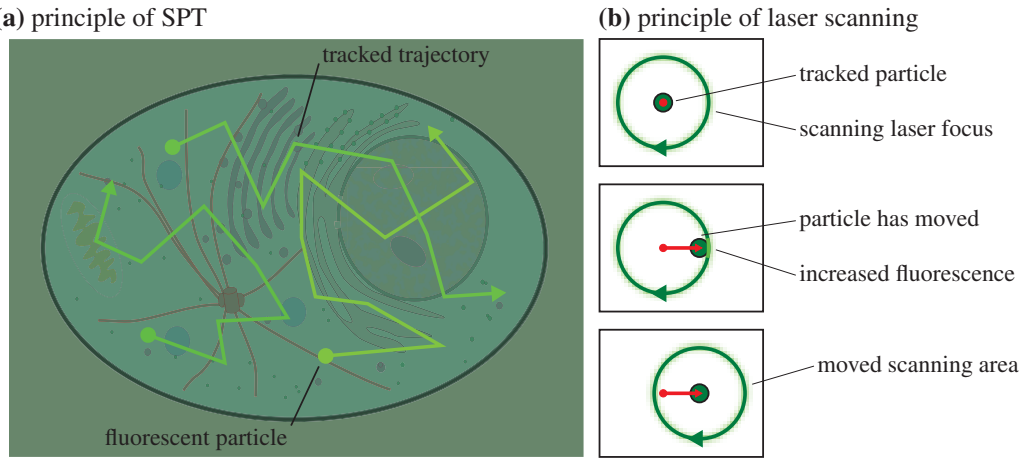


Figure 1.3.: Principle of single particle tracking (SPT). (a) Widefield view of the motion of particles within one plane of the cell. (b) SPT using a circularly scanning laser.

Observation of the transition into equilibrium

Instead of differentiating between single particles, other methods have been developed that rely on observing the transition of an entire system, *e.g.*, a cell, between a non-equilibrium and equilibrium state. This allows for measurements of the labeled particles at higher concentrations. In the simple case, these systems follow FICK'S second law (*cf.*, Equation (1.2)). As the fluorescence intensity is proportional to the particle concentration $I(\vec{r},t) \propto c(\vec{r},t)$ it can be observed easily with a microscope and the diffusion coefficient D can be extracted from the observed transition by solving FICK'S law.

One of these methods is fluorescence recovery after photobleaching (FRAP), which is illustrated in Figure 1.4. It relies on the artificial induction of a non-equilibrium state, by bleaching the fluorophores in a part of the sample (Figure 1.4b) [12, 150]. The particles in the space surrounding the bleached spot diffuse back into that area and the dark spot recovers (Figure 1.4c). The characteristic recovery time (Figure 1.4e) is inversely proportional to the mobility of the particles.

Fluorescence fluctuation techniques

The techniques described so far are either based on the measurement of single particles or perturbing the system into non-equilibrium and observing its relaxation. A different approach is to observe the system in its equilibrium state. At high temporal resolution, small fluctuations of the concentration $\delta c(\vec{r},t)$ around an average concentration $\langle c(\vec{r},t)\rangle_t$ become apparent for a reasonably small volume only (*cf.*, Figure 1.5):

$$c(\vec{r},t) = \langle c(\vec{r})\rangle_t + \delta c(\vec{r},t) \quad \text{with} \quad \langle \delta c(\vec{r},t)\rangle_t = 0. \tag{1.7}$$

The statistical fluctuations $\delta c\langle \vec{r},t\rangle$ are caused by inherent random processes that drive the motion of particles, *i.e.*, BROWNIAN motion. They can be measured by observing a small sub-volume of the sample, typically containing 10 to 100 particles. The statistical properties of $\delta c(\vec{r},t)$ are then used to draw conclusions about the underlying processes causing the motion. For example, the particle number $N(t) = c(t) \cdot V_{\text{obs}}$ in the observed sub-volume obeys POISSONIAN statistics:

$$\text{Var}\left(c(\vec{r},t)\right) = \left\langle \left[c(\vec{r},t) - \langle c(\vec{r})\rangle_t\right]^2 \right\rangle_t = \left\langle \delta c(\vec{r},t)^2 \right\rangle_t = \langle c(\vec{r},t)\rangle_t . \tag{1.8}$$

It states that the mean concentration of the particles $\langle c(\vec{r},t)\rangle_t$ can be extracted by analyzing the fluctuations, only.

Particles moving in and out of the observation volume create temporal correlations in $\delta c(\vec{r},t)$ on a time scale that is given by the average time the particle spends in the observation volume, *i.e.*, the diffusion time τ_D. τ_D is related to the size of the observation volume V_{obs} and the MSD of the particles. Assuming

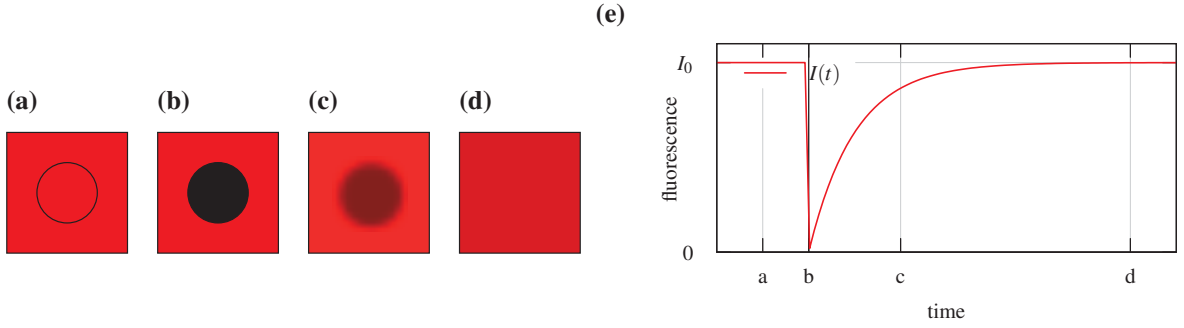


Figure 1.4.: Principle of fluorescence recovery after photobleaching (FRAP). (a) Initial equilibrium state. (b) Bleaching of the selected region. (c) Partial recovery of the sample. (d) Full recovery of equilibrium state. (e) Progression of the intensity.

a cubical or spherical volume with an edge length w_{obs}, $V_{\mathrm{obs}} \propto {w_{\mathrm{obs}}}^3$, the following scaling behavior can be derived by requiring $\mathrm{MSD}(\tau_D) \stackrel{!}{=} {w_{\mathrm{obs}}}^2$:

$$\tau_D \propto \frac{w_{\mathrm{obs}}^2}{D} \tag{1.9}$$

The correlation of $\delta c(\vec{r},t)$ can be extracted by an autocorrelation analysis of the measured intensity time-trace $I(\vec{r},t) \propto c(\vec{r},t)$. The normalized autocorrelation function is defined as:

$$G(\tau) = \frac{\langle \delta I(t) \cdot \delta I(t+\tau) \rangle_t}{\langle I(t) \rangle_t^2} = \frac{\langle I(t) \cdot I(t+\tau) \rangle_t}{\langle I(t) \rangle_t^2} - 1, \quad \tau > 0. \tag{1.10}$$

This method is called fluorescence correlation spectroscopy (FCS). It was first described in 1974 by Magde et al. [142, 143]. He also showed that

$$G(0) \propto 1/N, \tag{1.11}$$

where N is the average number of particles in the focus.

Nowadays FCS is typically implemented on a confocal microscope as shown in Figure 1.6a. Typically, the focal volume is $V_{\mathrm{obs}} = 0.2 \ldots 0.6\,\mu\mathrm{m}^3$ in size with a diameter of $w_{\mathrm{obs}} \approx 500\,\mathrm{nm}$ [179]. Such a small focal volume also allows for a measurement of the structure of the sample with a high spatial resolution, for example by scanning the sample point-wise. Single photon avalanche diodes (SPADs) or photomultiplier tubes (PMTs) are commonly used as detectors in confocal microscope setups. Fast readout electronics allow for temporal resolutions in the range of 10 ns, but pico-second time resolution can be achieved as well [228].

FCS is a common technique for the determination of particle concentrations and diffusion coefficients in solution. A review of the technique and its application is given in Ref. [118]. FCS is well established and has been used to determine concentrations and diffusion coefficients in various parts of the cell [13, 19, 21, 83, 133, 136]. It was further used to study active transport [112, 144], anomalous diffusion [192, 225, 232], and the internal motion of polymers [139, 197, 198, 240].

With fast detectors and acquisition electronics, high temporal resolution can be achieved, therefore FCS can be applied to also study photophysical effects of the dyes themselves. A common effect is blinking of fluorophores [82, 87, 234], which takes place on the µs to ms timescale. Also, the photon-antibunching of the fluorophores, *i.e.*, their fluorescence lifetime, can be measured.

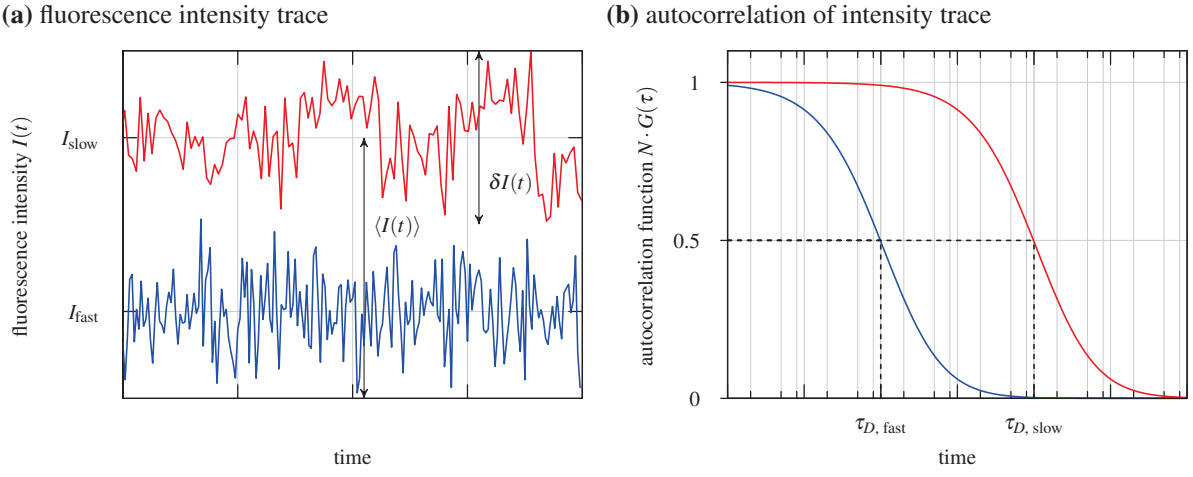


Figure 1.5.: Illustration of fluorescence correlation spectroscopy (FCS).

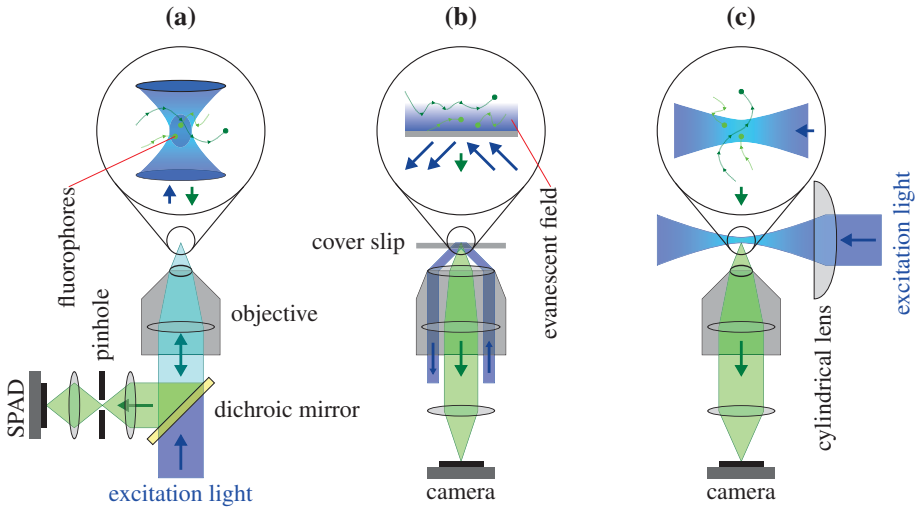


Figure 1.6.: Implementation of fluorescence correlation spectroscopy (FCS) on different types of microscopes. Excitation light is shown in blue, fluorescent light is shown in green. (a) Confocal fluorescence correlation spectroscopy (FCS). The additional pinhole in the detection path rejects out-of-focus light. (b) total internal reflection fluorescence correlation spectroscopy (TIRFCS). (c) lightsheet fluorescence microscopy (LSFM).

To further allow the observation of, for example, reaction kinetics, FCS can be extended to multiple spices labeled with differently colored dyes. The cross-correlation amplitude $G_{r,g}(0)$ of the two color channels, r and g (*e.g.*, 'red' and 'green'), is proportional to the concentration of constructs comprising both dyes. The normalized temporal cross-correlation function is defined as:

$$G_{r,g}(\tau) = \frac{\langle \delta I_r(t) \cdot \delta I_g(t+\tau) \rangle_t}{\langle I_r(t) \rangle_t \cdot \langle I_g(t) \rangle_t}, \quad \tau > 0. \tag{1.12}$$

2-color FCCS has been successfully used for various *in vivo* and *in vitro* measurements [13, 19, 83, 96, 134, 177, 220].

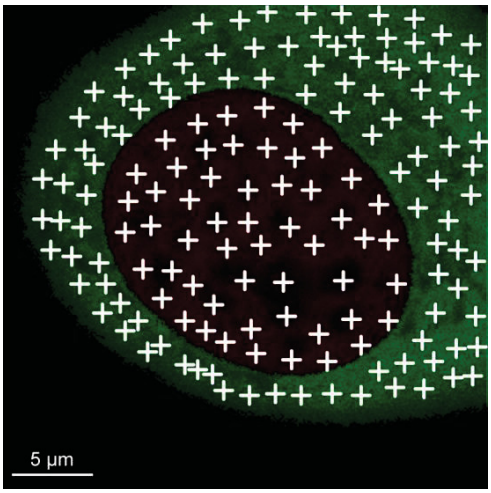
The FCS techniques described so far are all limited to single spot measurements. As the interior of cells is not uniform, FCS or fluorescence cross-correlation spectroscopy (FCCS) techniques have been extended to allow parallel multi-spot measurements. One possible solution is to carry out serial measurements at different positions, either at many positions and short dwell times [181], or small numbers of positions and long dwell times [57, 225]. By recording FCS data for many locations of the cell it is possible to generate two-dimensional distributions of particle numbers and diffusion coefficients. A typical map of diffusion coefficients is shown in Figure 1.7. The pixel dwell time strongly influences the noisiness of the autocorrelation curve (ACC) and thereby also the reliability of the extracted mobility parameters and concentration. In live cell measurements, the dwell time is usually limited by the cell which tends to move during acquisition and may also die if too much light is imposed.

Other techniques for measuring several positions that were implemented on confocal setups are based on scanning the laser focus over the sample, which can be done either circularly [182] or by following a line [176]. Both techniques allowed measuring of approximately 10 to 100 positions on a single trajectory.

Simultaneous measurements at different spots within the sample were achieved either with line confocal detection [22, 88], or spinning disk microscopy [156, 202]. Both techniques relied on cameras as fluorescence detectors. With at least 100 µs integration time for a single line, those cameras were several orders of magnitude slower than SPADs or PMTs, so temporal resolution is traded for spatial resolution.

For parallel multi-spot measurements SPAD arrays have been used to improve temporal resolution [40, 41, 74, 111]. These sensors combined 4 to 1024 individual SPADs on a single microchip, in different arrangements (2×2 or 1×8, up to 32×32). Their fast readout allows for temporal resolutions of 10 µs

(a) 158 single spot measurements



(b) reconstructed map of diffusion coefficients

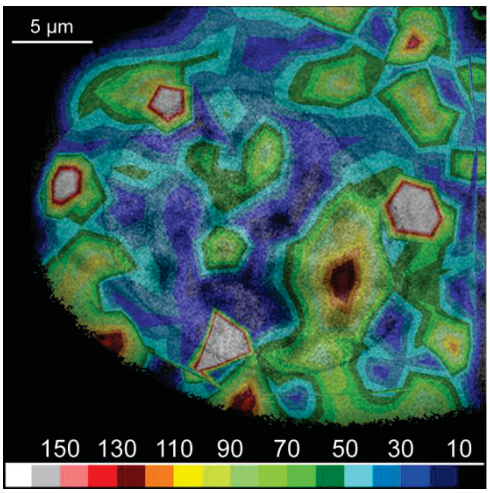


Figure 1.7.: Map of diffusion coefficients of two differently labeled molecules (green: eGFP, red: mRFP) in a HeLa cell. Diffusion coefficients are given in $\mu m^2/s$. Sample temperature was 37 °C. Images taken from Ref. [58].

or better. To create up to 1024 laser foci within the sample simultaneously, spatial light modulator (SLM) based on liquid crystal displays (LCDs) or diffractive optical elements (DOEs) were used. The fluorescence from each focus is then imaged onto a single SPAD. Recently (2013), such a setup was successfully used for live-cell imaging [111].

If more than a single spot is measured simultaneously, not only the signal autocorrelation or cross-correlation between two species sharing the same focal volume can be calculated, but also the cross-correlation between two spatially distinct spots. In comparison to an autocorrelation analysis, a spatial cross-correlation analysis implies a certain averaging over space, as more than a single spot is taken into account. For two foci at positions $\vec{r}$ and $\vec{r}+\vec{s}$ the spatial cross-correlation function (CCF) is defined as

$$g(\tau,\vec{s}) = \frac{\langle \delta I(\vec{r},t) \cdot \delta I(\vec{r}+\vec{s},t+\tau)\rangle_t}{\langle I(\vec{r},t)\rangle_t \cdot \langle I(\vec{r}+\vec{s},t+\tau)\rangle_t}. \tag{1.13}$$

A major advantage of cross-correlation is that the effect of the size and shape of the focal volume is less important than its distance, which is typically known precisely. Therefore, diffusion measurements based on spatial cross-correlation have been successfully carried out in various cellular systems [53, 54, 84, 114, 115, 236–238].

An alternative approach for parallel multi-spot FCS was taken by Kannan et al. [107] in 2007. Combining a wide-field microscope and a sensitive camera, they pioneered imaging fluorescence correlation spectroscopy (imaging FCS) on a total internal reflection (TIRF) microscope. The setup is illustrated in Figure 1.6b. A thin layer above the glass cover slip is illuminated by an evanescent wave. This limits the observation volume to a 100 nm to 200 nm thin layer above the cover slip which results in a fluorescence signal with very low background contributions that often renders the actual signal undetectable. The penetration depth is very small, so this technique only allows to observe outer structures of a sample. Such a setup was successfully used to analyze diffusion in biological membranes with relatively slow diffusing molecules (τ_D in the order of milliseconds) [15–18, 24, 78, 185].

In 2010, Wohland et al. [241] pioneered the use of a lightsheet fluorescence microscopy (LSFM) to perform imaging FCS. The working principle of a LSFM is shown in Figure 1.6c [75, 95]. A thin sheet of light with a diameter of about $1-2\,\mu m$, typically formed by a cylindrical lens, illuminates a slice of a sample. The fluorescence is detected perpendicular to the illumination and projected onto a fast image sensor. At each pixel a volume of approximately $1-2\,\mu m^3$ is observed. In contrast to

TIRF, this technique is not limited to the surface of a sample. The setup is less complex compared to multi-focal setups and much easier to operate. The technique is usually called SPIM-FCS (selective plane illumination microscopy fluorescence correlation spectroscopy).

Similar to confocal setups, SPIM-FCS can be extended to SPIM-FCCS by detecting in two channels simultaneously. It was successfully used to study protein interactions [25, 34].

1.1.3. Correlation analysis

Signal correlation is usually performed by a so called correlator. Being either a piece of software or dedicated hardware, the correlator computes Equation (1.12). To cover a large range of different delay values τ, also referred to as lag-times, ranging from microseconds to seconds, most correlators use a logarithmic scaling [187]. Some dedicated hardware correlators also feature data acquisition circuitry and perform the correlation analysis in real time [62]. Commercially available hardware correlators provide up to 32 channels [5, 46] which are sufficient for a small number of pixels [74]. They are typically based on reconfigurable logic, such as field programmable gate arrays (FPGAs), or dedicated application-specific integrated circuits (ASICs) [62, 63, 91]. Custom built hardware correlators are typically FPGA-based [72, 73, 93, 101, 102, 105, 135, 256].

Most software implementations are optimized to process photon arrival times [128, 191, 226], where the detector attaches a high-resolution time-tag to each photon event. Even parallel approaches exist for detectors with multiple pixels [49, 129, 217]. Recently, first graphics processing unit (GPU)-based implementations were reported [127, 222]. A more detailed overview of various correlators is given in section 3.2.

1.2. Aim of this thesis and outline

As depicted above, fluorescence correlation spectroscopy (FCS) and its two-dimensional advancement imaging fluorescence correlation spectroscopy (imaging FCS) are versatile methods to gain information about the properties of small molecules in living cells. With the availability of new single photon avalanche diode (SPAD) arrays with a camera-like amount of pixels and frame rates above 100 000 fps, the question arises whether these devices are suitable as detectors in LSFM.

In this work, the performance of two different SPAD arrays, the *RADHARD2* (*cf.*, section 2.1.3) and the CHSPAD (Swiss single photon avalanche diode array, see section 2.1.4), both designed in EDOARDO CHARBON'S lab (TU Delft, Netherlands and EPFL, Switzerland), were evaluated with respect to their usefulness for SPIM-FCS. Further, the feasibility of such detectors to create real-time diffusion maps was assessed.

A first step was to design a readout for both detectors, to allow image acquisition in real time at highest frame rates (*cf.*, section 4.1.1 and section 4.2). Secondly, the correlation analysis of the raw image data had to be implemented on high performance computing (HPC) platforms to achieve real-time image processing. This was done with the example of two recent CPU architectures, AMD *Piledriver* and INTEL *Haswell* (*cf.*, section 5.3), a consumer grade GPU, an NVIDIA *GTX 780 Ti* (*cf.*, section 5.4), and a XILINX VIRTEX 2 field programmable gate array (FPGA) (*cf.*, section 5.5). Moreover, considerations regarding FPGA-based dataflow computing were made (*cf.*, section 5.5.7). Finally, the performance of the SPAD arrays integrated into a custom build SPIM was evaluated (*cf.*, section 6). For that purpose, several fluorescent dyes were tested and, as a last step, first *in vivo* measurements of enhanced green fluorescent protein (eGFP) oligomers in HeLa cells are shown (*cf.*, section 6.9.3).

1.3. List of publications

Several aspects of this thesis work have already been published, particularly the design of FPGA-based hardware correlators, which is also described in section 5.5:

[152] Gábor Mocsár, Balázs Kreith, **Jan Buchholz**, Jan Wolfgang Krieger, Jörg Langowski, and György Vámosi. Note: Multiplexed multiple-tau auto- and cross-correlators on a single field programmable gate array. *Review of Scientific Instruments*, 83(4):046101, 2012. doi: 10.1063/1.3700810.

[155] **Jan Buchholz**, Jan Wolfgang Krieger, Gábor Mocsár, Balázs Kreith, Edoardo Charbon, György Vámosi, Udo Kebschull, and Jörg Langowski. FPGA implementation of a 32x32 autocorrelator array for analysis of fast image series. *Optics Express*, 20(16):17767, 2012. doi: 10.1364/OE.20.017767.

An in-depth comparison of different cameras for imaging FCS was published as:

[201] Anand Pratap Singh, Jan Wolfgang Krieger, **Jan Buchholz**, Edoardo Charbon, Jörg Langowski, and Thorsten Wohland. The performance of 2D array detectors for light sheet based fluorescence correlation spectroscopy. *Optics Express*, 21(7):8652, 2013. doi: 10.1364/OE.21.008652.

Further details on the data evaluation and experiment control software *QuickFit3*, which was used in this thesis (*cf.*, section 2.3.3), are assembled in:

[121] Jan W. Krieger, **Jan Buchholz**, Nicolas Schnellbächer, Christoph S. Garbe, and Jörg Langowski. QuickFit 3.0 – a data evaluation package for (imaging) fluorescence correlation spectroscopy. *BMC Bioinformatics*, 2015. in preparation.

The experience gained in parallelization and vectorization of correlators was used to accelerate a kinetics simulation:

[153] Norbert Mücke, Stefan Winheim, Holger Merlitz, **Jan Buchholz**, Jörg Langowski, and Harald Herrmann. Kinetics of intermediate filament assembly in vitro: A monte carlo simulation study. In preparation, 2015.

2. Basic principles

2.1. Photodetectors

2.1.1. Photon counting Detectors

In this section, first the theory of operation of different photodetectors is described. The second part gives a more detailed description of the two detectors, the Swiss single photon avalanche diode array (CHSPAD) and the *RADHARD2*, which were used in the course of this thesis.

Single photon avalanche diodes

Single photon avalanche diodes (SPADs) are the main workhorse for single photon counting applications like fluorescence correlation spectroscopy (FCS). These detectors can be seen as a solid state equivalent to photomultiplier tubes, which convert incoming photons into electrical pulses that can be counted. Their ability to count individual photons mitigates against gain noise or circuit noise. SPADs are photodetectors operated above the breakdown voltage V_{break}, the so called GEIGER-mode. This is done by applying a reversed bias operating voltage V_{OP} well above the breakdown voltage V_{break}. A single incident photon creates an avalanche, which can be detected and counted easily. When operated below break-down voltage, the avalanche effect leads to a stochastic gain, which is proportional to the photocurrent.

Figure 2.1a shows the cross-section of a typical reach-through $p^+ - \pi - p - n^+$ APD structure used as SPAD. It is built from four differently doped regions, two low doped areas π and p, and two highly doped regions p^+ and n^+. Photon absorption occurs in the relatively large π region. A photon hitting the area creates an electron-hole pair within the low doped absorption area π of the diode with a certain probability. The single charge-carrier injected into the absorption area is then amplified in the bias field into an electron avalanche, the so called avalanche breakdown which is shown in Figure 2.1b. These avalanches can be detected as a strong current spike [47, 86, 184].

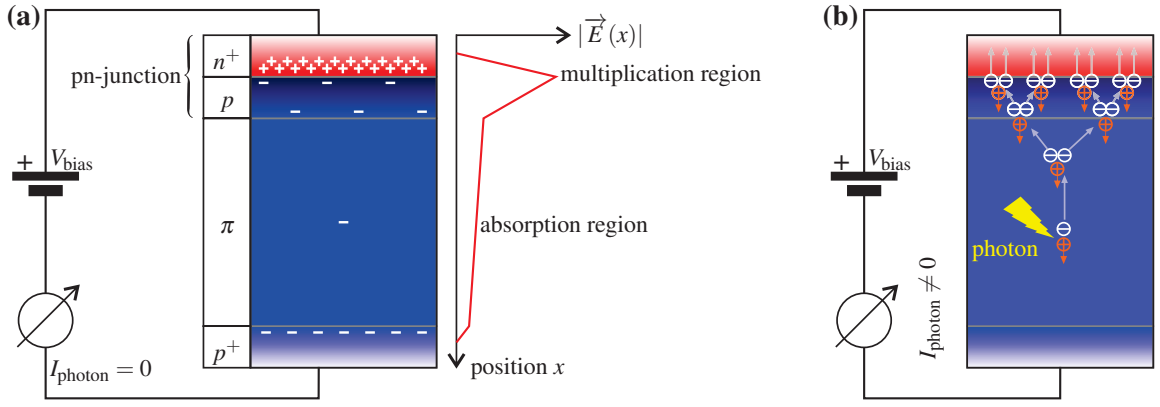


Figure 2.1.: Schematic drawing of a single photon avalanche diode (a) Cross-section of a reach-through $p^+ - \pi - p - n^+$ APD structure used as SPAD including space charge distribution under reverse voltage. The plot on the right shows the strength of the electric field. The p^+ and n^+ regions are heavily doped. (b) Charge carrier multiplication in a SPAD under reverse voltage. Image adapted from Ref. [119].

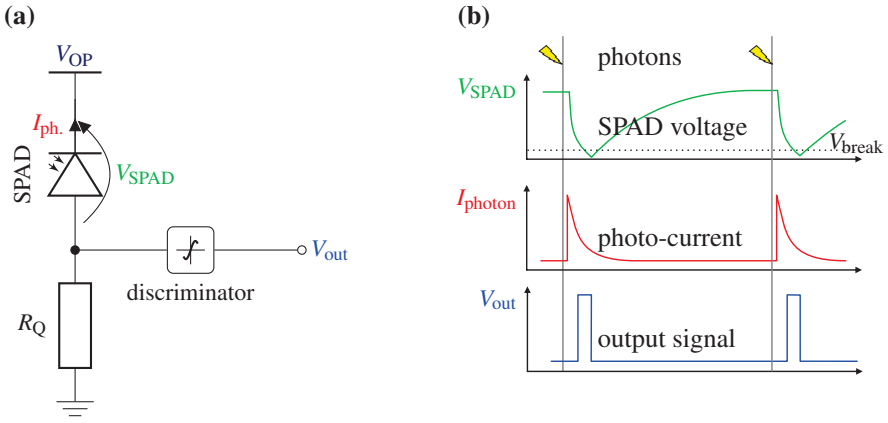


Figure 2.2.: Passive quenching circuit of a single photon avalanche diode. (a) Passive quenching circuit for a SPAD with a series-connected resistor. (b) Course of voltages and currents in that circuit. Images adapted from Ref. [119].

The drifting process is driven by V_{OP}, and limited by collisions of the charge carriers with the semiconductor lattice. When the electrons reach the multiplication region, a thin $p-n^+$ junction with a high electric field $|\vec{E}(x)|$, an avalanche with millions of secondary electrons is created by repeated impact ionization.

In case of APDs, amplification increases with the reverse bias V_{OP}. If V_{OP} rises above the breakdown voltage V_{break}, the amplification gets virtually infinite. At that point, photons create self-sustaining avalanches, and the avalanche photodiode (APD) operates in photon counting or GEIGER mode. Shortly after a photon hits the detector, the current rises with the starting of the avalanche and causes the resistance across the SPAD to drop. By series connecting the SPAD with a resistor, the breakdown of V_{SPAD} can be detected by a discriminator circuit (this can be seen in Figure 2.2a).

Each avalanche must be stopped, so called quenched, to avoid damage to the diode due to the current and to re-arm the detector. Generally, two implementations are possible [47]:

- **active quenching:** By adding a dedicated circuit that detects an avalanche and actively lowers V_{SPAD}, the avalanche is stopped.

- **passive quenching:** A resistor is placed in series with the SPAD. If the current increases through the diode, a higher voltage drops over the resistor that lowers the diode voltage V_{SPAD} until $V_{SPAD} < V_{break}$ and the avalanche stops. This circuit is shown in Figure 2.2a, the voltage curve and the progression of the currents are shown in Figure 2.2b.

Independent of whether the active or passive quenching technique is used, the reverse bias voltage of the SPAD is reduced below the breakdown voltage. This inactivates the SPAD, as now the voltage across the diode is too low to cause another avalanche and to detect any photons. This "dead time" (typically in the order of 100 ns), can be reduced by shortening R_Q [132, 159]. This technique is called **active recharge**.

Feedback of the detector leads to additional pulses that were not caused by actual photon detection This so called **afterpulsing** may occur when charge carriers get trapped in the depletion region and get released after a short random period when the SPAD is again in operation. These carriers then re-trigger an avalanche, which was not caused by an actual photon. Studies showed, that the afterpulsing follows a power-law timing distribution [99, 119, 219, 258] (see section 2.4.3 for further details).

Sensor	array size	pixel type	pixel pitch d_{pixel} [µm]	SPAD diameter d_{SPAD} [µm]	fill factor FF [%]
Rech et al. [175]	8×1	fully addressable	250	50	N/A
Niclass et al. [157]	32×32	single addressable	58	7	1.1
Carrara et al. [36]	32×32	memory	30	6	3.1
Burri et al. [31]	128×512	memory	24	6	4.9
Niclass et al. [160]	128×128	column TDC	25	7	6
Gersbach et al. [69]	32×32	TDC	50	5.5	1
Tisa et al. [218]	32×32	TDC/memory	100	20	3.1

Table 2.1.: Comparison of recent SPAD arrays with their most relevant properties.

Single photon avalanche diode arrays

In the last decade, arrays of SPADs have been developed to allow camera like single-photon imaging at high frame rates [20, 30, 32, 35, 69, 70, 160, 193, 218]. These sensors consist of up to 65 536 single SPADs organized in a pixel array structure, as known from conventional cameras. They are typically based on complementary metal-oxide-semiconductor (CMOS) chips. Some devices also integrate additional quenching or readout circuitry, partially on a per-pixel basis. A comparison of the major properties is shown in Table 2.1.

The least complex devices are linear SPAD arrays with diodes that are fully accessible on the outside. Such an array, comprising eight pixels, has been demonstrated in Refs. [97, 175]. Quenching and readout was performed on external electronics.

In parallel, two dimensional detectors were developed. Although the pixels are rectangular, the SPADs are typically circular in these devices due to the high electric field strengths. With a higher number of single SPADs, full access to every diode was not possible any more. Pixels became row-accessible, with shared column outputs that are typically connected to data processing electronics, *e.g.* field programmable gate arrays (FPGAs). The first larger array of that kind was published in 2005 [157]. Each of the 32×32 pixels contained an integrated discriminator and a column output driver implemented as five transistors. Only one pixel could be addressed and read simultaneously (*cf.*, Figure 2.3c for the structure of the pixels).

The sensors described above were not suitable for the read-out of many pixels at high temporal resolutions. Either only a single line of SPADs was available or the pixel had to be selected prior to a measurement. In the latter case, scanning was employed to simulate imaging. To allow imaging with a large amount of pixels simultaneously, all pixels have to be scanned. Therefore, the information that a photon was detected, must be conserved until the pixel can be read.

The active area of a SPAD, *i.e.*, the region of the pixel which actually detects the photons, is crucial for single photon detectors. To keep the circuit area in a pixel at a minimum and therefore allow for larger SPADs, mostly single bit memories were implemented [30, 32, 35]. In existing SPAD arrays this requires 12 or more transistors. The saved information only tells whether none or at least one photon was detected (*cf.*, Figure 2.3d for pixel structure). The probability to miss photon events when more than one photon is detected by the SPAD can be reduced by increasing the readout speed. A more detailed description of the implementation of two such sensors given in section 2.1.3 and section 2.1.4. Data from such arrays is typically read row-wise. When a row has been read, the state of the memories is reset and the next row is accessed. This mode of operation is called “rolling shutter” mode and also common in CMOS cameras (see below).

A major drawback of the read-out procedure described above is that the temporal resolution (*i.e.*, the accuracy of the detection of photon arrival times) now fully relies on the time it takes to read the entire image. By only reading a subset of pixels, the temporal resolution can be improved, but then the high

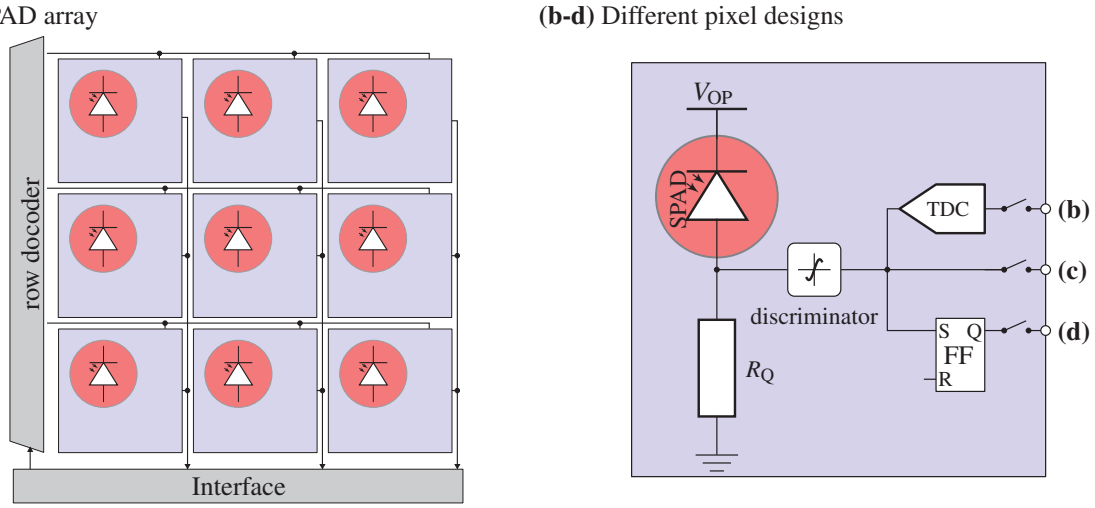


Figure 2.3.: SPAD array with different pixel designs. (a) Row-wise accessible SPAD array. (b) A TDC is embedded in each pixel to resolve the timing of a photon event. Typically the photon time-tag is saved per pixel for later readout. (c) Each pixel is individually addressable, no further circuitry. (d) A latch stores a photon event in each pixel. An additional reset signal is required. The switches at the output of each pixel are controlled by the row-decoder.

pixel count is not used any more. To overcome this limitation, time to digital converters (TDCs) were integrated into each column [160] or each pixel [69] (*cf.*, Figure 2.3b for pixel structure). In the latter case, fill-factor was traded for additional circuitry. For common circular SPADs with a diameter d_{SPAD} and quadratic pixels with an edge length of d_{pixel}, the fill-factor is defined as:

$$\mathrm{FF}_{\mathrm{pixel}} := \frac{\text{active area}}{\text{pixel area}} = \frac{\pi \cdot (d_{\mathrm{SPAD}}/2)^2}{d_{\mathrm{pixel}}^2}. \tag{2.1}$$

A possible solution to overcome the typically low fill-factors in most available SPAD arrays, is to attach microlenses to the pixels. These concentrate the incident light onto the active area of the SPADs. See section 4.3.7 for more details.

The readout of the in-pixel TDCs is typically done in rolling-shutter mode, too, but now time-stamps are read from every pixel. This requires higher bandwidths, as several bit have to be read from each pixel. Such arrays achieve temporal resolutions in the order of 100 ps.

Tisa et al. [218] reported a sensor with a TDC in every pixel. This TDC can also be reconfigured to operate as an 8 bit counter to allow camera-like imaging. SPAD arrays, whether they are based on TDCs or not, typically allow for read-out rates of 100 kHz to 150 kHz for full frames of 32×32 to 512×128 pixels.

2.1.2. Linear detectors

Introduction

The detectors described so far generate a series of pulses representing a single, detected photon each. In the following, "linear" detectors will be described. These detectors generate a signal, typically a charge that is transformed to a voltage, that is proportional to the number of incident photons.

CMOS cameras

CMOS cameras are made of a two-dimensional array of pixels. Each pixel consist of a photodiode and an additional transistor used as source follower (see Figure 2.4). This transistor amplifies the signal from

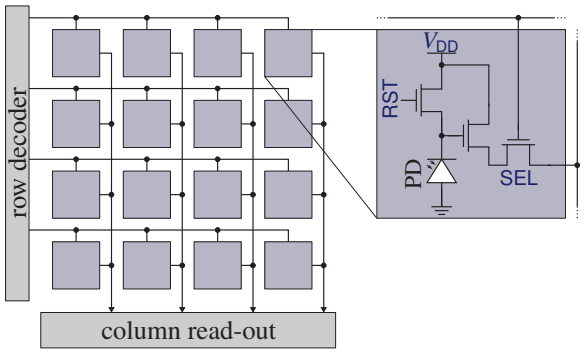


Figure 2.4.: CMOS active pixel sensor. The inset shows the simplified electronic circuit of a single pixel with three transistors (3T). Figure adapted from Ref. [165].

the photocharges in the photodiode, making the pixel 'active'. A specific pixel can be selected by driving the selection transistor (SEL). A third transistor used for reset (RST). Due to the active nature of the pixels, such devices are also called active pixel sensor (APS). The digitalization is done in each column by a column amplifier. Typically, pixels are addressed row-wise.

Additional transistors can be used to separate photodetection and photoconversion regions to eliminate specific noise. As already described for SPAD arrays, also in CMOS image sensor, the space in each pixel has to be shared between the electronics and the photodiode. The reduced fill-factor is often compensated by applying microlenses to the pixels. Another solution is to use back-illuminated sensors. Here the substrate is thinned down so that light can penetrate the sensor from the back. Other possibilities are to use transparent substrates (*e.g.*, silicon-on-sapphire).

Light detection in a CMOS-pixel works as follows: Prior to integration, the diode is reset to a distinct voltage ($V_{\text{diode}} = V_{\text{DD}} - V_{\text{transistor, threshold}}$) by turning on the reset transistor for a short time. After that the diode is electrically floating and photogenerated carriers accumulate in the diode. The accumulated charges lower the potential of the diode, which is then amplified by the electronics in the pixel. After a certain integration time, the pixel is read and the operation starts over. Reading a pixel's value does not destroy the charge of the diode, because of the high input impedance of the amplifier. Generally, two different modes for reading full images exists:

- **rolling shutter:** The sensor is continuously read line by line. When a line has been read, the pixels are reset. In this mode, the integration time of each line starts at a different point in time.
- **global shutter:** All pixels are reset simultaneously and the integration time is identically. During readout, the detector is insensitive ("dead-time"). For this mode, usually one capacitor is added to each pixel. An additional transistor is used to transfer the charges of the diode to the capacitor, which is then read-out via the amplification transistor.

The analogue output of the pixels is digitized by an analogue-to-digital converter (ADC) and the values are typically transferred to a computer for further processing. Depending on the size of the array and the required frame rate, CMOS cameras either use a single ADC per chip, or in the other extreme, they use one ADC per column.

Scientific CMOS cameras

Recently, a new generation of CMOS cameras was introduced, which is especially designed for scientific image acquisition. These scientific CMOS (sCMOS) cameras offer more than a million pixels ($d_{\text{pixel}} = 6.5\,\mu\text{m}$), very low noise, and a frame rate in the region of 100 fps. Some sCMOS sensors further allow for global shutter modes (with an addition transistor per pixel, 5T). Usually microlenses are applied to

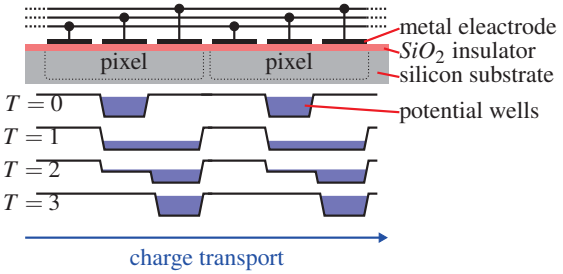


Figure 2.5.: Design and shift operation of a three-phase CCD. A pixel is defined by a group of three gates. Charges are accumulated in potential wall below the center gate. Readout is done from left to right.

the sensors to reach a photon detection probability (PDP) above 60 %. Such cameras are optimized for low-light applications, for example fluorescence microscopy and bio-imaging [8, 39].

CCD cameras

A different type of analog linear detectors are charge coupled device (CCD) cameras. Contrary to CMOS cameras, these are made of passive pixels. CCD cameras are basically a two-dimensional array of metal-insulator-semiconductor (MIS) capacitors (see Figure 2.5). They consist of a piece of bulk silicon with an insulation layer and metal gates on top. By applying a specific voltage pattern to those gates, a potential well is formed below the electrodes in the bulk silicon. Photoinduced charges accumulate in that well below the electrode of the pixel. By applying different electric patterns to the gates, accumulated charge carriers can be transported across the array and towards the readout registers. From there they are finally transported off the chip. Further details are given in Refs. [89, 208].

The simplest CCD sensors are organized as arrays of light sensitive pixels only. This is shown in Figure 2.6a. For faster readout, such sensors are subdivided into several regions. The actual readout is done by shifting the charges vertically towards the readout registers and then horizontally to external amplification and ADC electronics. A major drawback of this scheme is the effect of smearing that can occur, since the pixels are still sensitive during the charge transport phase. This problem is addressed by interline transfer or frame transfer sensors. Here, the photocharges are transferred to light-shielded

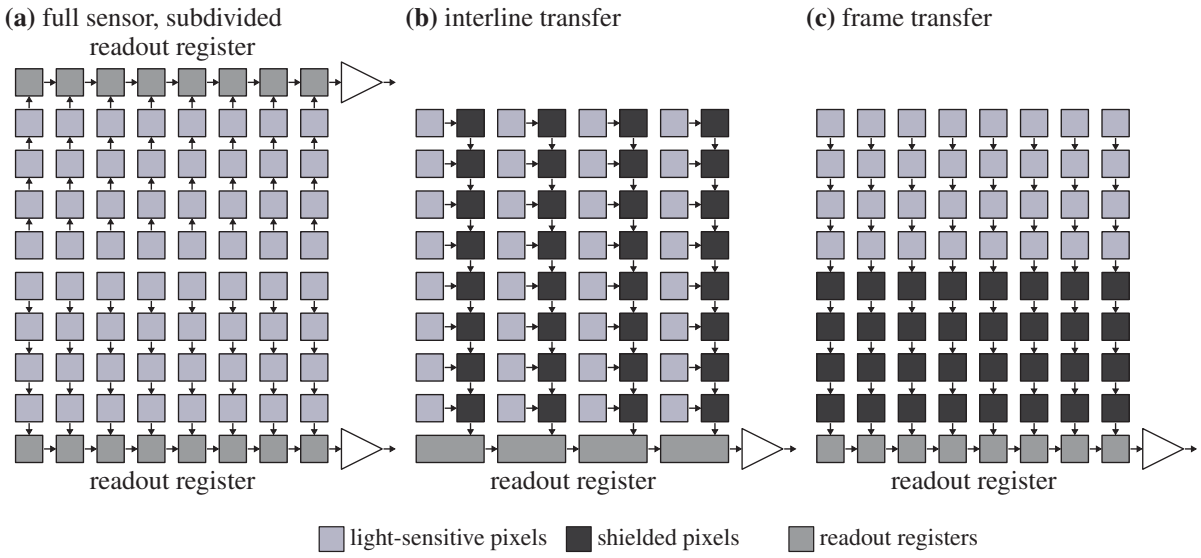


Figure 2.6.: Different transfer architectures of CCD sensors. Figures adapted from Refs. [89, 208].

areas prior to readout. In the case of an interline transfer sensor, which is shown in Figure 2.6b, the pixel content is first shifted horizontally into the shielded shift registers. For readout, charges are then shifted vertically into the main readout register. Figure 2.6c shows a frame transfer sensor, here the half of the sensor is shielded. During readout, the image is first transferred to the shielded area, which can be done very fast and reduces the effect of smearing. The latter two methods allow for parallel acquisition during readout, which reduces the dead-time of the detector.

EMCCD cameras

To further increase the signal to noise ratio (SNR) for very low light levels, additional on-chip amplification is done. This is achieved by placing a high-voltage gain-register between the readout register and the output amplifier. Figure 2.7 shows the design of such a sensor. At each stage of that shift register, the transported electrons are multiplied by impact ionization similar to the avalanche effect in APDs. Such a sensor is therefore called electron multiplying charge coupled device (EMCCD). Although the amplification in each single stage of the shift register is low, a large number of stages (in the order of 100) gives EM-gains of up to $1000\times$ [10].

Quantum efficiency and photon detection probability

The quantum efficiency (QE), η_{det}, is a crucial property of an image sensor especially when used in low-light applications. It is defined as the ratio of incident photons to converted electrons, or pulses in case of digital counters. The photon detection probability (PDP) is defined as the effective quantum efficiency, which also considers the fill-factor FF (equation (2.1)):

$$\text{PDP} = \eta_{\text{det}} \cdot \text{FF}. \tag{2.2}$$

Figure 2.8 gives an overview of typical QEs of various cameras as a function of the wavelength of the incident light. The figure includes front- and back-illuminated cameras as well as SPAD arrays. Back-illuminated cameras can reach QEs above 90 %. In case of the SPAD arrays, only the active area is taken into account. Experiments that were carried out in the course of this thesis were done in the blue to green spectral range ($\lambda = 450\,\text{nm} \ldots 550\,\text{nm}$). This is also the part of the spectrum where all sensor have their maximum.

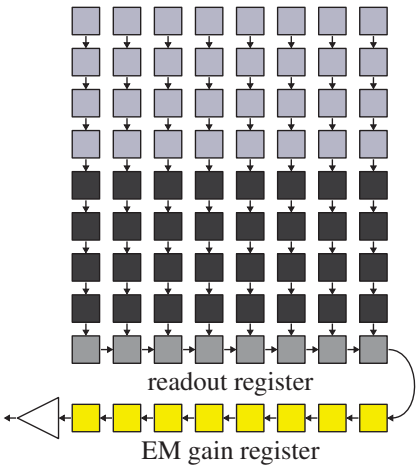


Figure 2.7.: Architecture of a frame-transfer EMCCD image sensor. EM gain registers are shown in yellow.

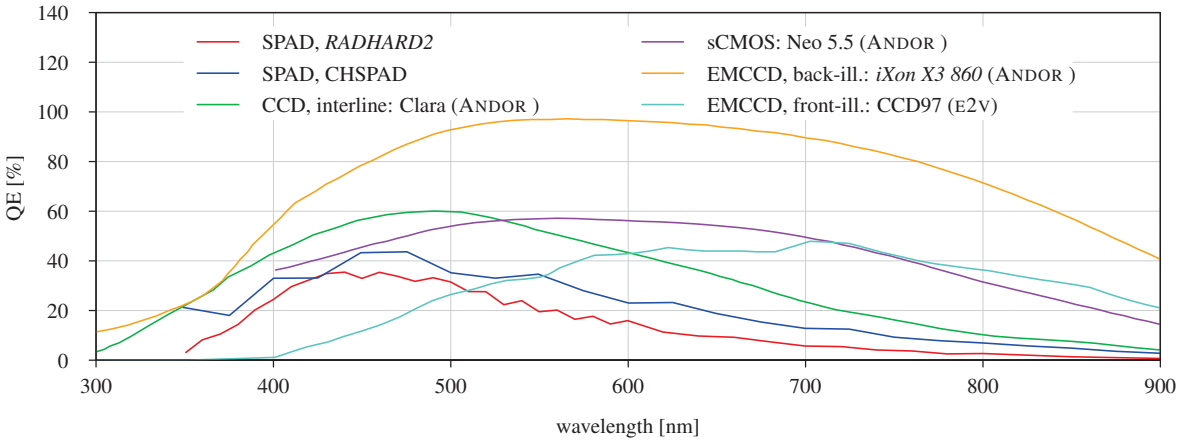


Figure 2.8.: Quantum efficiency of various scientific cameras. All cameras except the ANDOR *iXon X3 860* are front-illuminated. In case of the SPAD arrays, only the active area is taken into account. Excess bias voltage for the SPAD arrays: $V_{\mathrm{ex},\,RADHARD2} = 3.3\,\mathrm{V}$ and $V_{\mathrm{ex},\,\mathrm{CHSPAD}} = 4.0\,\mathrm{V}$. Data taken from Refs. [8–10, 33, 158, 193].

Statistics of image sensors

In photodetectors, the number of actually detected photons N_{photon} obeys POISSONIAN statistics and the variance equals the mean number of photons:

$$\mathrm{Var}(N_{\mathrm{photon}}) = \left\langle N_{\mathrm{photon}} \right\rangle . \tag{2.3}$$

The mean number of actually detected photons is defined as

$$\left\langle N_{\mathrm{photon}} \right\rangle = \eta_{\mathrm{det}} \cdot \Phi \cdot \Delta t_{\mathrm{exp}}, \tag{2.4}$$

for an integration time Δt_{exp} and a mean photon flux Φ.

Photodetectors also tend to have a certain dark count rate (DCR), which is introduced by random output pulses that do not originate from incident photons. Main causes for these pulses are avalanches that are started by thermally created fluctuations or afterpulsing. The DCR $\dot{N}_{\mathrm{dark}}$ can be reduced by cooling the sensor. The rate fall exponentially with the temperature.

The rare occurrences of dark counts can be characterized by a POISSONIAN distribution with a mean value of $N_{\mathrm{dark}} = \dot{N}_{\mathrm{dark}} \cdot \Delta t_{\mathrm{exp}}$ and a variance of $\sigma^2_{\mathrm{dark}} = N_{\mathrm{dark}}$. That said, equation (2.4) and equation (2.3) can be extended to

$$\left\langle N_{\mathrm{photon,dark}} \right\rangle = \left(\eta_{\mathrm{det}} \cdot \Phi + \dot{N}_{\mathrm{dark}}\right) \cdot \Delta t_{\mathrm{exp}} \tag{2.5}$$

$$\mathrm{Var}(N_{\mathrm{photon,dark}}) = \mathrm{Var}(N_{\mathrm{photon}}) + \sigma^2_{\mathrm{dark}} = \left\langle N_{\mathrm{photon,dark}} \right\rangle . \tag{2.6}$$

For a linear detector, the average number of detected photoelectrons can be written analogously to equation (2.4):

$$\langle N_{\mathrm{e}} \rangle = \langle G \rangle \cdot \eta_{\mathrm{det}} \cdot \Phi \cdot \Delta t_{\mathrm{exp}}, \tag{2.7}$$

where $\langle G \rangle$ is the average charge amplification factor for on-chip amplification (*i.e.*, for EMCCD cameras). As the stochastic processes of photodetection and gain are statistically independent, the variance of the detected photoelectrons is given as:

$$\mathrm{Var}(N_{\mathrm{e}}) = \left\langle N_{\mathrm{e}}^2 \right\rangle - \langle N_{\mathrm{e}} \rangle^2 = \eta_{\mathrm{det}} \Phi \cdot \Delta t_{\mathrm{exp}} \cdot \langle G \rangle^2 . \tag{2.8}$$

The conversion of the analogue output signal of the chip is done by a linear analogue-to-digital converter (ADC). The ADC is used to quantize the signal into $2^{\mathscr{R}_{\mathrm{ADC}}}$ different values, where $\mathscr{R}_{\mathrm{ADC}}$ is the

integer resolution in bits. The digital output of the ADC is commonly called analog-to-digital units (ADUs). The conversion process is modeled by a conversion factor $\mathscr{A}_{\mathrm{ADC}}$ that is given in analog-to-digital units (ADUs) per photoelectron.

Even in full darkness, linear image sensors tend to exhibit a small background signal, which has several reasons:

- **dark current:** Within each active pixel of the detector, thermally created electron-hole pairs may be generated that are indistinguishable from photogenerated charges. Their contribution (see above) to the image and background noise is:

$$\langle N_{\mathrm{dark}} \rangle = \langle G \rangle \cdot \dot{N}_{\mathrm{dark}} \cdot \Delta t_{\mathrm{exp}}, \quad \sigma^2_{\mathrm{dark}} = N_{\mathrm{dark}} \tag{2.9}$$

- **clock-induced charges:** In CCD cameras, the shifting process might induce an average of N_{cic} additional charges per readout cycle. This effect can be reduced by a careful design of the readout. Due to the POISSONIAN nature of the process, it contributes as follows to the noise of the sensor:

$$\sigma^2_{\mathrm{cic}} = \langle G \rangle^2 \cdot \langle N_{\mathrm{cic}} \rangle . \tag{2.10}$$

- **read noise:** The read noise is not signal-dependent, but imposed by the readout electronics (*e.g.*, amplifiers or ADCs):

$$\langle N_{\mathrm{read}} \rangle = 0, \quad \sigma^2_{\mathrm{read}}. \tag{2.11}$$

- **quantization noise:** The quantization noise describes the noise imposed by the limited resolution of the ADC. Its upper limit is typically the least significant bit:

$$\langle N_{\mathrm{ADC}} \rangle = 0, \quad \sigma^2_{\mathrm{ADC}} \sim \frac{1}{\mathscr{A}_{\mathrm{ADC}}}. \tag{2.12}$$

All the noise components described above typically contribute to the background image noise:

$$\sigma^2_{\mathrm{back}} = \sigma^2_{\mathrm{dark}} + \sigma^2_{\mathrm{cic}} + \sigma^2_{\mathrm{read}} + \sigma^2_{\mathrm{ADC}} \sim \sigma^2_{\mathrm{dark}} + \sigma^2_{\mathrm{read}}. \tag{2.13}$$

2.1.3. The *RADHARD2* single photon avalanche detector array

The first sensor that was evaluated for selective plane illumination microscopy (SPIM)-FCS in the course of this thesis was the *RADHARD2* SPAD array, designed in EDOARDO CHARBON'S lab. A micrograph of the chip is shown in Figure 2.9. Table 2.2 summarizes a subset of signals that were driven by the controller described in section 4.1. Figure 2.10 shows a schematic drawing of a single pixel.

The name stems from its radiation tolerance and the sensor can resist up to 30 mRad [35]. A drawback of the radiation hardness is the decreased fill-factor and an increase in electronics size. In total a single pixel comprises 12 transistors. The chip is fabricated in a 0.35 µm high voltage CMOS process. Each SPAD is operating in GEIGER mode above breakdown voltage that was $V_{\mathrm{break}} = 18.8\,\mathrm{V}$ for the described sensor. Operating voltage was $V_{\mathrm{OP}} = 22.0\,\mathrm{V}$, so the excess voltage was $V_{\mathrm{ex}} = 3.2\,\mathrm{V}$ accordingly.

The *RADHARD2* SPAD array comprises 32×32 pixels, that are accessed row-wise. Each pixel is $30\,\mu\mathrm{m} \times 30\,\mu\mathrm{m}$ in size with an active area of the SPAD $d_{\mathrm{SPAD}} < 6\,\mu\mathrm{m}$ (based on personal communication, 4 µm is more likely). In total, the fill factor is 1.4 %.

Driven at almost 100 MHz, full frames integrations times as low as 1.2 µs can be achieved. A very low afterpulsing probability $< 1\,\%$ and a reasonable dark-count rate of 140 Hz at room temperature and $V_{\mathrm{ex}} = 3.3\,\mathrm{V}$ were reported. The active area of the SPAD has a PDP of about 35 % in the blue to green range, which is ideal for several commonly used biological dyes (for a comparison of the PDP of different cameras see Figure 2.8). When considering fill-factor and the PDP of the active area, a per-pixel PDP of 0.5 % is reached.

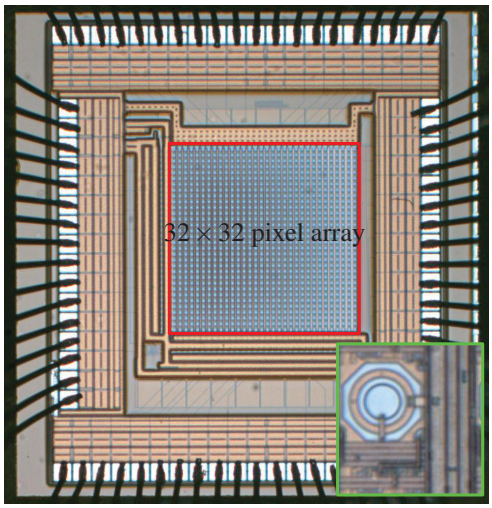


Figure 2.9.: *RADHARD2*: micrography of the chip. The green inset shows a single pixel. The active area is the inner blue circle. Image adapted from Ref. [35].

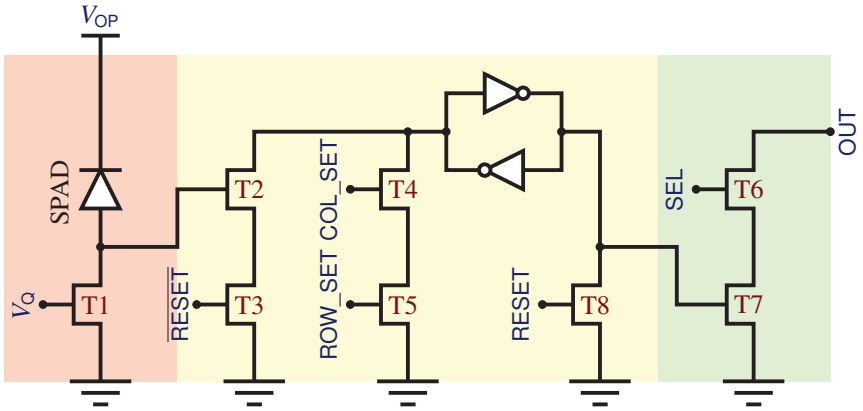


Figure 2.10.: *RADHARD2*: Transistor level diagram of a single pixel's electronics. The circuitry consists of three major parts: detection (red), memory (yellow) and output (green). Transistor T1 is used as tunable quenching resistors (via V_Q). All signals shown (RESET, ROW_SET, COL_SET and SEL) are generated internally by the sensor. The outputs OUT of all pixels in a column are connected and are driven by a pull-up. Schematic adapted from Ref. [35].

signal / pin	description
V_{DD}	positive logic supply voltage, provided by motherboard, 3.3 V
V_{OP}	positive SPAD supply voltage, 22.0 V
V_Q	tunable passive quenching resistor, 1.0 V
ROW_ADD (5 bit)	address of row to select (0..31)
ROW_CLK	rising-edge latches ROW_ADD
ROW_RESET	reset currently selected row
DATA_OUT (32 bit)	output of the selected row

Table 2.2.: *RADHARD2*: input voltages and signals.

Principle of operation. When a photon hits the diode, it creates an avalanche that causes a current flow through the diode and the quenching resistor T1. T2 senses the avalanche voltage and causes the latch to be set to logic 1. The latch is reset by driving T8. T3 prevents memory conflicts if the SPAD is firing during reset.

The avalanche stops automatically when the voltage drops below V_{break} (passive quenching, *cf.*, section 2.1.1). This characteristic can be fine-tuned by setting an appropriate bias voltage V_{Q} selecting a trade-off between dead time and afterpulsing probability.

2.1.4. The Swiss single photon avalanche diode array

In this section, the second evaluated SPAD array is described. The design of the SPAD itself is based on the design of the *RADHARD2* SPAD array. Some details presented in here are based on personal communication with the designer SAMUEL BURRI (EPFL, Switzerland). Further description of the sensor can be found in Refs. [29, 31, 33]. Table 2.3 compares several properties of the two sensors, the *RADHARD2* and the CHSPAD.

Figure 2.11 shows a micrograph of the CHSPAD. The mounting of the CHSPAD within the inner part of the microscope is shown in Figure 2.14 (see section 2.3.1 for details on the microscope). Figure 2.12 shows a circuit diagram of the internal layout of the CHSPAD. Every four columns share a single output pin, which is selected by COL_ADD. Due to this inner structure, reading a single row (512 pixels) takes four clock cycles.

Circuitry of a single pixel. A single pixel's circuitry, shown in Figure 2.13, consists of 14 transistors which can be grouped into three major parts. Each pixel uses four voltages and seven signals (five global and two local). Table 2.4 summarizes the voltages and signals of a single pixel. The detection part (red) comprises the large SPAD, a tunable resistor T12 used for passive quenching, a transistor T2 for active recharge, an off-switch T1, a gating-transistor T4, and a transistor T3 to set the memory (yellow). Therein the event of a photon is saved in a simple N-type metal-oxide-semiconductor (NMOS)-latch (T7 and T8, with tunable resistors T5 and T6, via V_{Top}). The state is reset by T9 with an internal reset signal generated when a new row is selected during readout. Finally, the third part of the circuitry is the output drivers (made-up of T10 and T11) which drive the output low when the pixel is selected via SEL and a photon was detected. In each column the outputs of all pixels are connected and are driven by a pull-up.

Row selection is done by driving ROW_ADD (*cf.*, Figure 2.12; see Figure 4.1 for a timing diagram). The address is internally latched with a rising edge of ROW_CLK. The same applied for the addressing of the columns.

property	unit	*RADHARD2*	CHSPAD
manufacturing process		0.35 µm CMOS	
array size	pixels	32×32	512×128
single pixel size	µm^2	30×30	24×24
diameter of active area d_{SPAD}	µm	< 6	< 6
fill-factor	percent	3	5
quantum efficiency	%	35	45 ($\lambda = 450$ nm)
PDP	%	0.5	2.2
DCR	Hz	140	366
quenching transistor bias voltage V_{Q}	V	1.0	1.1‡
SPAD operating voltage V_{OP}	V	22.1	24‡
SPAD excess voltage V_{ex}	V	3.2	3.7‡

Table 2.3.: Comparison of the *RADHARD2* SPAD array and the CHSPAD array. Data from Ref. [33, 35]. Values for the CHSPAD are given for $V_{\text{ex}} = 4$ V. ‡ Data is based on evaluations described in section 4.3.

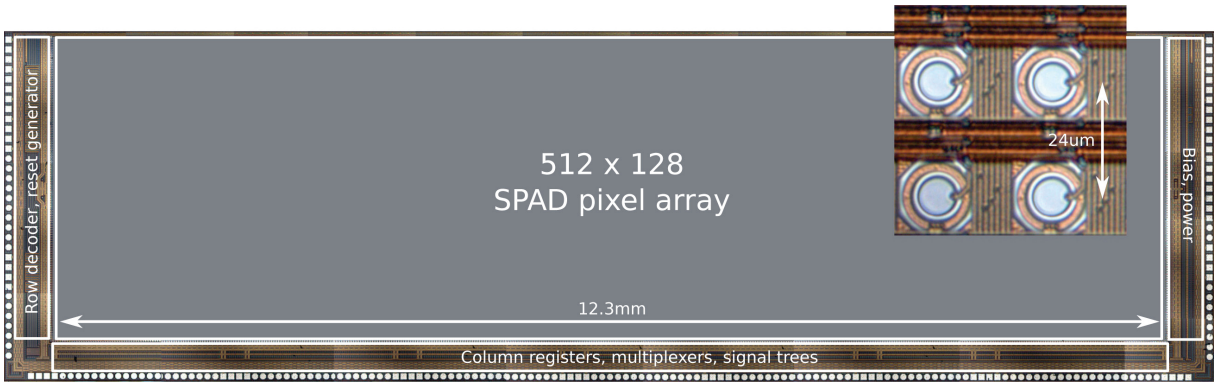


Figure 2.11.: Micrograph of the CHSPAD. The pixel area is enclosed by logic on three sides. Two sensors can be put together with a gap of less than 6 pixels. The small inset shows 2×2 pixels with a pitch of 24 µm. The SPAD is the blueish circular area surrounded by the reddish guard ring. Image taken from Ref. [33].

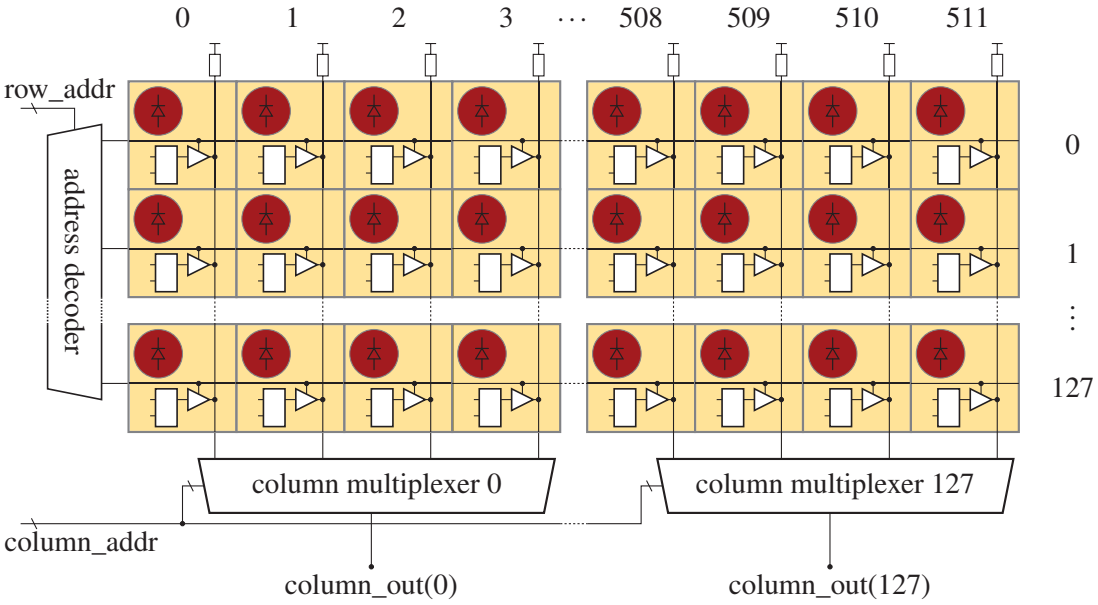


Figure 2.12.: Schematic drawing of the internal layout of the CHSPAD. The array is internally organized in 128 rows and 512 columns. The outputs of all pixels are connected column-wise with a pull-up circuit. For each output pin, the output of one of four neighboring columns can be selected via column_addr. All signals are connected to an FPGA.

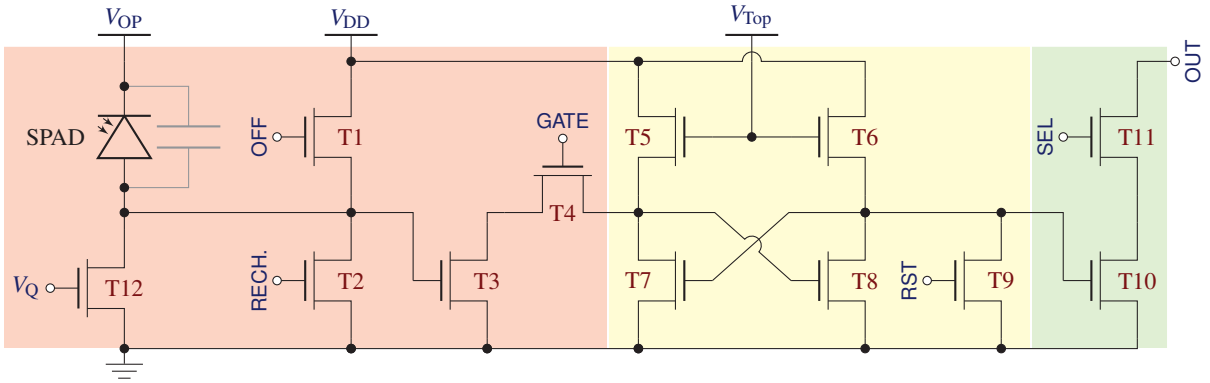


Figure 2.13.: CHSPAD: Transistor level diagram of a single pixel's electronics. The circuitry consists of three major parts: detection (red), memory (yellow) and output (green). Three of the 12 transistors (T5, T6 and T12) are used as tunable resistors (via V_Q and V_{Top}). Two further unused transistors are not shown. Additionally to the logic supply voltage V_{DD} a second positive supply voltage V_{OP} is needed for the SPAD. Two of the five input signals (RST and SEL) are generated internally by the sensor, the others (RECHARGE, OFF and GATE) apply for the entire sensor. The outputs OUT of all pixels in a column are connected and are driven by a pull-up. Schematic adapted from Ref. [33].

signal / pin	description
V_{DD}	positive logic supply voltage, 3.3 V
V_{OP}	positive SPAD supply voltage, 20 V to 25 V
V_Q	tunable passive quenching resistor, 0 V to 1.2 V
V_{Top}	tunable SRAM resistors, 2.0 V
(SPAD)OFF	disable SPAD, must not be active when RECHARGE is active
RECHARGE	force recharge of SPAD, bypass passive quenching
GATE	disable photon event recording in memory
ROW_ADD 7 bit	address of row to select (0..127)
ROW_CLK	rising-edge latches ROW_ADD
COL_ADD (1..0)	address for multiplexer of column output (0..3)
COL_CLK	rising-edge latches COL_ADD
COL_OUT (127..0)	output of the selected output columns
PULLUP	enable pull-up of the output columns
RESET_EN	enable reset of row after deselection
ROW_SEL_EN	enable row selection by ROW_ADD
TCSPC (not shown)	TCSPC-mode, bypass transistor between output of memory and gate of T10
TUPC (not shown)	TUPC-mode, bypass transistor between output of SPAD and gate of T10

Table 2.4.: CHSPAD: input voltages and signals.

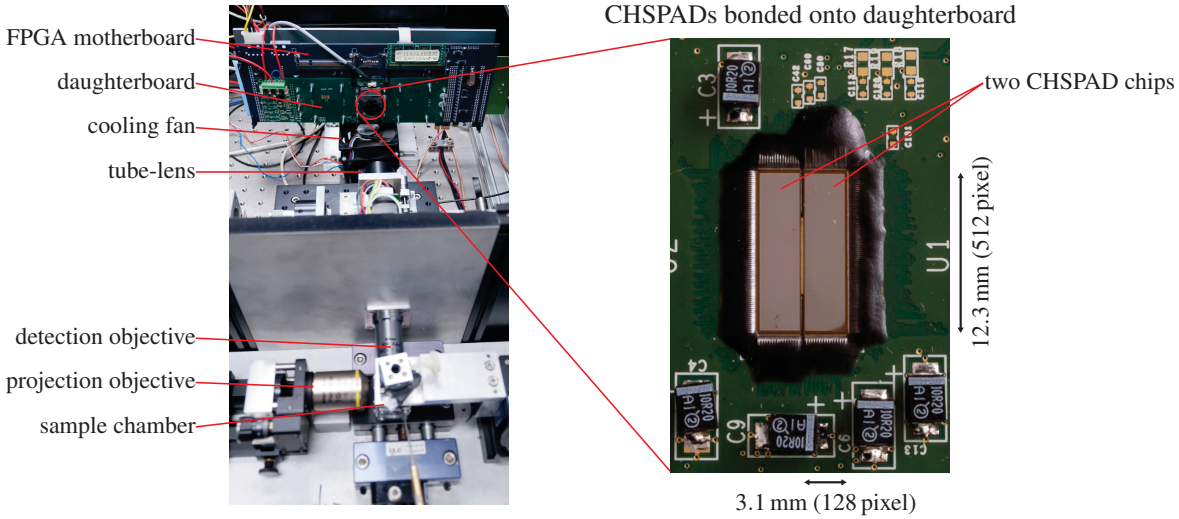


Figure 2.14.: CHSPAD: Sensor including hardware platform mounted in the inner part of the SPIM setup.

Photon detection. Within the active SPAD an avalanche is triggered when a photon hits the active area. The current flow raises the anode voltage and naturally quenches the avalanche. A higher voltage at the gate of T3 causes the latch to flip if T4 is open. For re-arming the SPAD after a break-through two different possibilities exist. Either the anode voltage can be lowered again by an active recharge pulse via T2, or passively by the tunable resistor T12. The SPAD can be disabled by driving T1, which then lowers the excess bias voltage of the diode by raising the anode voltage.

2.2. Fluorescent labels

In this thesis, the motion of molecules is measured with the help of fluorescent labels. This way, it is possible to observe or track a single species of interest, even in a mixture of various particles (*e.g.*, in a cell). In the following, the basic principles of fluorescence are introduced. Section 2.2.4 gives and overview of the dyes, which were used in this work.

2.2.1. Fluorescence

Fluorescence is the emission of a photon by a molecule after its excitation by an incident photon. This is illustrated by the JABLONSKY diagram, as shown in Figure 2.15. The Energy spectrum is split into quantized electronic states S_ν of a specific energy E_ν, numbered by the quantum number ν. S_0 is called the electronic ground state, $S_1, S_2, \ldots$ are called excited states. Each of these electronic states further split up into multiple vibrational and rotational sub-states (S^*_ν), that only differ slightly in their energy level. The more complex a fluorescent molecule is, the more densely these states are packed, almost forming a continuum. Vibrational relaxation involves dissipation of energy to the surrounding and thus cannot occur for isolated molecules.

A photon of the wavelength $\lambda_{\text{ex.}}$ carries the energy of

$$E_{\text{photon}} = \frac{h \cdot c_0}{\lambda_{\text{ex.}}}, \tag{2.14}$$

where h is PLANCK'S constant and c_0 is the speed of light in vacuum. If the photon's energy matches the difference of an excited state (in S_μ, $\mu = 1, 2, \ldots$) and the state of an electron within S^0, the electron may absorb the photon's energy and gets excited into a higher state. Typically, the electron gets excited into a vibrational sub-state, and decays quickly to the ground-state of the respective electronic state by heat

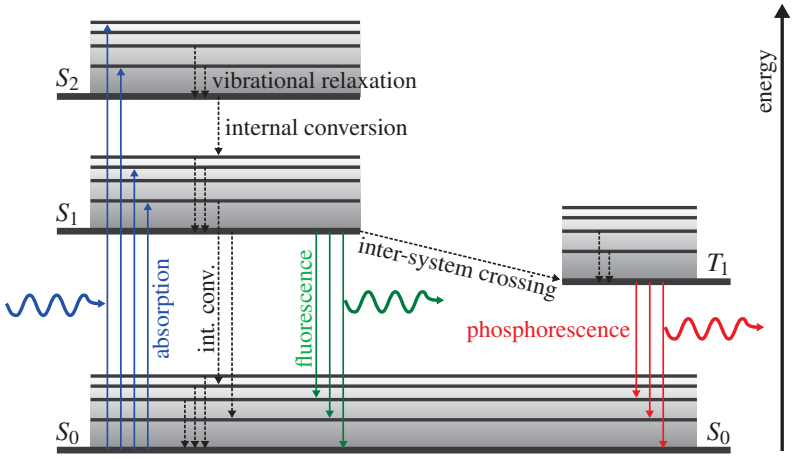


Figure 2.15.: JABLONSKI diagram of a fluorescent molecule. S_ν and T_ν denote the singlet and the triplet state of the molecule with the quantum number ν. Solid lines indicate transitions between electronic states with absorption or emission of a photon (wavy line). Non-radiative transitions are shown as black dashed lines. The vibrational ground state of each electronic state is indicated with a thick line.

dissipation. After a certain time, the electron decays back into S_0^*. The energy difference is emitted as a new photon with an increased wavelength $\lambda_{\text{fluorescence}} > \lambda_{\text{excitation}}$. Once again, the electron decays into ground state S_0. The difference between the peak excitation wavelength and the fluorescence wavelength $\Delta\lambda = \lambda_{\text{fl.}} - \lambda_{\text{ex.}}$ is called STOKES shift. The energy difference is typically lost in heat dissipation. The timescale of the absorption is typically in the picosecond range, whereas the internal conversion is on the femtosecond timescale. The fluorescent lifetime is of the order of nanoseconds.

So far only the excitation and decay within the same spin-multiplicity (*e.g.*, singlet to triplet) was considered. So called inter-system crossings, the transitions between states of different spin multiplicity can give rise to the effect of phosphorescence. Here, a classically forbidden transition from, for example, the lowest triplet state to the singlet ground state occurs. Still statistically unfavored, these decays are allowed in quantum mechanics. Thus, typical lifetimes of such triplet states are of the order of milliseconds to minutes.

2.2.2. Photobleaching

Photobleaching is another decay channel for excited fluorescent molecules. Instead of a reversible absorption and emission of photons, the incident light induces a photochemical destruction of the molecule. As this process typically involves a structural change, photobleaching is mostly non-reversible or very long-lived.

2.2.3. Brightness of a fluorophore

The molar extinction coefficient $\varepsilon_{\text{fluor}}$ is a measure of how strong a molecular species absorbs light. The actual absorbency A depends on the pathlength l, $\varepsilon_{\text{fluor}}$, and the concentration c (BEER-LAMBERT law):

$$A = \varepsilon_{\text{fluor}} \cdot l \cdot c. \tag{2.15}$$

The quantum yield ϕ_{fluor} of a fluorescent molecule is the ratio between the numbers of emitted and absorbed photons:

$$\phi_{\text{fluor}} = \frac{N_{\text{emitted}}}{N_{\text{absorbed}}}. \tag{2.16}$$

The brightness of a fluorescent molecule is proportional to the product of the extinction coefficient and the quantum yield:

$$B_{\text{fluor}} \sim \varepsilon_{\text{fluor}} \cdot \phi_{\text{fluor}} \tag{2.17}$$

2.2.4. Dyes

Throughout this work, four different types of fluorescent dyes were used, which are described in the following. Table 2.5 summarizes their fluorescent properties.

Fluorescent micro-spheres

Calibrating and characterizing the microscope requires for bright probes. For that purpose, two different types of fluorescent beads were used, *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads (*T7279*, INVITROGEN) and *FluoSpheres YG* $\varnothing = 100\,\text{nm}$ fluorescent beads (*F8803*, INVITROGEN). These are polystyrene beads that are stained throughout with fluorescent dyes.

Figure 2.16a shows an example image of *TetraSpeck* beads for different excitation wavelengths. In case of the *TetraSpeck* beads, four different fluorescent dyes are embedded, for the yellow-green (YG) beads, a single dye is used with a peak absorption at $\lambda = 505\,\text{nm}$. Both samples are well suited for 488 nm laser excitation. The brightness of the *FluoSpheres YG beads* is equivalent to 7.4×10^3 fluorescein molecules [215].

TetraSpeck beads have a diffusion coefficient of $D_{\text{TS},\vartheta=20\,^\circ\text{C}} = (3.9 \pm 0.6)\,\mu\text{m}^2/\text{s}$ in water [154], and a diameter of (100 ± 7) nm [214]. The same values were assumed for the YG beads.

Quantum dots

Quantum dots (QDs) are another frequently used sample in fluorescence microscopy. They are made of nano-crystals of semiconductor materials (roughly 10^4 atoms). Dimensions are so small ($\sim 6\,\text{nm}$, the range of the DE BROGLIE wavelength of an electron at room temperature), that the charge carriers (electrons and electron holes) within crystal undergo quantum-mechanical effects and energy can only exist at discrete levels. Examples of the photoluminescence of different quantum dots is shown in Figure 2.16b. The properties of such crystals (*e.g.*, color) can be be fine tuned (*e.g.* by changing the size, or the chemical composition). To use the crystals in biological samples, they are coted with a polymer and can be further coupled to proteins [212]. In contrast to other dyes, QDs do not bleach even after hours of excitation [204].

The size of the *QDot-525 streptavidin ITK* (*Q10041MP*, INVITROGEN) used in here is comparable to a large macromolecule or protein (roughly 15 nm to 20 nm) and is therefore much smaller as the $\varnothing = 100\,\text{nm}$ fluorescent beads described above. Thus, they are much better suited for performance evaluation of a microscopic setup for the evaluation of small molecules. Another major advantage of the quantum dots is their broad extinction coefficient in the blue spectrum [213].

dye	$\varepsilon_{\text{fluor}}$ [1/(M cm)]	ϕ_{fluor}	$B_{\text{fluor}} = \varepsilon_{\text{fluor}} \cdot \phi_{\text{fluor}}$ [1/(mM cm)]
Alexa-488	73.000	0.92	67
eGFP	55.000	0.60	33
QDot-525 streptavidin ITK	130.000	0.29	38
fluorescein	76.900	0.93	72
TetraSpeck $\varnothing = 100\,\text{nm}$ fluorescent beads	N/A		
FluoSpheres YG $\varnothing = 100\,\text{nm}$ fluorescent beads	7.4×10^3 fluorescein equivalents		

Table 2.5.: Properties of the dyes. Data was taken from Refs. [146, 203, 210, 211, 216].

Alexa-488

A third type of fluorescent dyes, a chemical fluorophore, that was used in this thesis is *Alexa-488*. The chemical structure is shown in Figure 2.17a. *Alexa-488* has its absorption maximum at 495 nm and its emission maximum at 519 nm [209]. With these properties the dye is, as the name indicates, well suited to be used with 488 nm lasers. The quantum yield is $\phi = 0.92$ and the molecular extinction coefficient $\varepsilon = 73\,000/(\mathrm{cm\,M})$ [210, 211]. In comparison to the formerly used standard dyes (fluorescein, rhodamine, etc.), *Alexa-488* dyes tend to be more photostable.

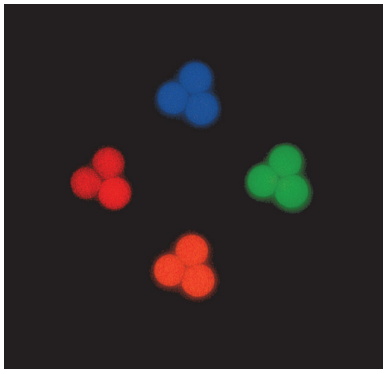
Due to its small size, the diameter is about 1 nm, the fluorescent molecule is relatively fast ($D_{20,\mathrm{W}} = 406\,959\,\mu\mathrm{m}^2/\mathrm{s}$) [173]. Its high mobility makes it hard to be characterized with CMOS based cameras. Typically, the molecule is used as a label and is connected to some other molecule at the marked position (* in Section 2.2.4). Furthermore, *Alexa-488* is used as a standard for calibration in confocal FCS [59].

Green fluorescent protein

The green fluorescent protein (GFP) is a protein composed of 238 amino acids (26.9 kDa) that exhibits bright green fluorescence when exposed to blue or ultraviolet light. GFP was first isolated from jellyfish *A. victoria* in 1962 [196]. Its discovery and development was honored with the Nobel Prize in chemistry in 2008. The enhanced form of GFP that was used in here, enhanced green fluorescent protein (eGFP) [257], has a fluorescence quantum yield of $\phi = 0.60$ and an extinction coefficient of $\varepsilon = 55.000/(\mathrm{M\,cm})$[146]. Instead of the rather complex excitation spectra of wild type GFP, eGFP has a single excitation peak at 488 nm. In cell and molecular biology, GFP is often used as a marker for gene expression or as a marker molecule [38]. Via transfection methods, the GFP-DNA can be incorporated into a target cell.

In this work, different eGFP oligomers were used as samples (either in solution of in live cells). As an example, the commonly assumed form eGFP tetramers is shown in Figure 2.17b. The proteins in solution were extract from transient (non permanent) transfected cells expressing the specific constructs.

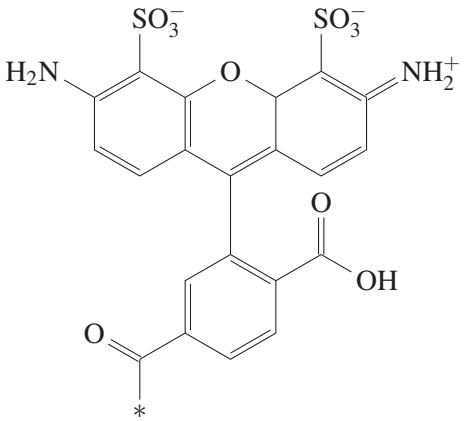
(a) *TetraSpeck* beads



(b) *QDot-525*

Figure 2.16.: *TetraSpeck* beads and *QDot-525*. (a) Four exposures of three *TetraSpeck* beads. Image taken from `https://www.lifetechnologies.com/order/catalog/product/T7279`. (b) Photoluminescence of $\varnothing = 6\,\mathrm{nm}$ quantum dots of different chemical compositions. Image taken from `http://www.sigmaaldrich.com/materials-science/nanomaterials/quantum-dots.html`.

(a) *Alexa-488*



(b) eGFP oligomers

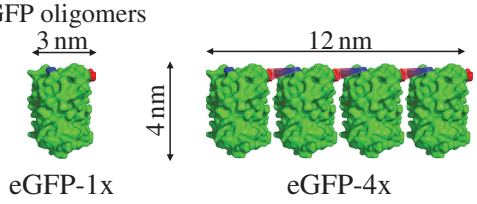


Figure 2.17.: *Alexa-488* and eGFP oligomers. (a) Structure of the *Alexa-488* fluorophore. Image adapted from `http://www.operon.com/products/custom_oligos/modifications/fluorescentdyes.aspx`. (b) eGFP monomers and eGFP tetrameters as used as inert tracer molecules. Crystal structure taken from Ref. [167]. Images adapted from Ref. [119].

2.3. Fluorescence microscopy

2.3.1. SPIM

In the following, the principle of lightsheet fluorescence microscopy (LSFM) using the example of selective plane illumination microscopy (SPIM) is described. The second part introduces the microscope, that was used in the course of this work.

Principle of lightsheet fluorescence microscopy

Confocal microscopy, as shown in Figure 2.18b and also Figure 1.6, offers a high resolution and a fast readout but lacks the ability for parallel multi-spot measurements. Figure 2.18c shows the principle of lightsheet fluorescence microscopy (LSFM). A cylindrical lens is used to focus down an expanded beam of parallel light in only one direction. By that, a thin sheet of light, the lightsheet is created. Perpendicular to the lightsheet an infinity corrected detection objected is positioned. Its object plane is superimposed by the lightsheet. The thickness of the lightsheet and the z-extend of the image plane of the detection objective are typically matched to avoid out-of-focus light.

Being a wide-field illumination method (*cf.*, Figure 2.18a) combined with a strongly improved z-sectioning, it can be seen as compromise between confocal and epi-fluorescence microscopy. In comparison to wide-field fluorescence microscopy, a major advantage of lightsheet microscopy is its ability to reject out-of-focus light in the detection path. Furthermore, the selective illumination reduces negative photophysical effects on the sample as, for example, photobleaching of the fluorophores or photodamage [95]. As other fluorescence microscopes, lightsheet microscopy allows for using multiple colors illumination and detection.

History

The technique of lightsheet microscopy (LSM) was first described by Siedentopf and Zsigmondy [199] in 1902. First, this technique was used to study scattered light from nano-sized particles [145]. In 1925 Zsigmondy [260] received the Nobel prize.

First fluorescence measurements were published in 1993 by Voie et al. [223]. Their setup made use of a cylindrical lens only and achieved a minimum width of the lightsheet of 20 µm. A breakthrough was achieved in by Huisken et al. [95] in 2004 who replaced the single cylindrical lens used for lightsheet formation by a combination of a cylindrical lens (CL) and a projection objective (PO, *cf.*, Figure 2.20). This type of setup is referred to as the first selective plane illumination microscopy (SPIM) setup. A major advantage of such a setup is, that the optical parameters of the lightsheet do not longer depend on the quality of the cylindrical lens, but on the projection objective, which are typically available in higher

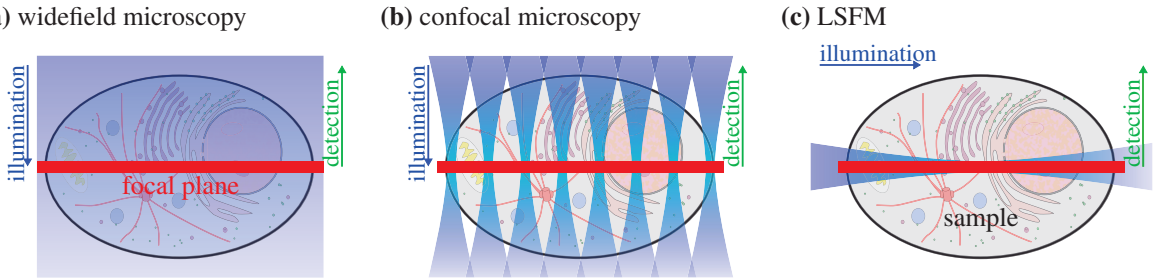


Figure 2.18.: Comparison of different fluorescence microscopy techniques.

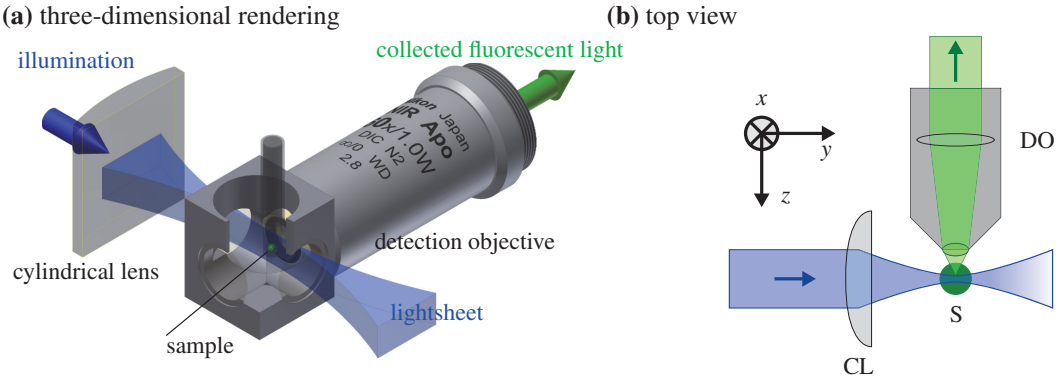


Figure 2.19.: Principle of LSFM. The three-dimensional rendering was provided by JAN W. KRIEGER.

quality [75]. Instead of using a cylindrical lens, a sheet of light can also be formed with a scanning laser [108].

Description of the setup

The setup used in the course of this thesis is based on this design proposed by Greger et al. [75]. It was designed and built by JAN W. KRIEGER (DKFZ Heidelberg, Germany) as an enhanced version for cameras with a pixel size of 20 µm to 30 µm [119].

Figure 2.20 shows a schematic drawing of the setup, a photograph is shown in Figure 2.21*. As a light source, a $\lambda = 488\,\text{nm}$ diode laser is used with an output power of 60 mW. Its emission light is cleaned up by a bandpass filter (CF). An additional neutral density filter is built into the beam path to attenuate the laser power. That was crucial as the laser beam is more stable when the laser is driven at its specific output power. A custom built laser shutter is the next part within the light path. After that, the beam is expanded (BE). The gimbal mounted mirror (GMM) in combination with the relay telescope, that is formed by the lens-doublets L1 and L2, allows for fine adjustment of the z-position of the lightsheet.

Viewed from top, the incident laser beam is not affected by the cylindrical lens and focused down to a thin sheet of light by the projection objective. Viewed from the side, the cylindrical lens (CL) and the projection objective (PO) form a telescope producing a parallel beam within the sample chamber (SC). First described in Ref. [75], this method allows using high-quality microscope objective lenses to form the lightsheet. Instead of a single cylindrical lens this results in less optical aberrations and a overall thinner lightsheet.

As projection objective a 10× microscope objective is used (numerical aperture of 0.3 and a focal length of $f_{\text{PO}} = 20\,\text{mm}$). The lightsheet geometry can be approximated with a GAUSSIAN beam with a width $w_{\text{LS}}(y)$ along the propagation direction (y-axis) given by [119, 183]:

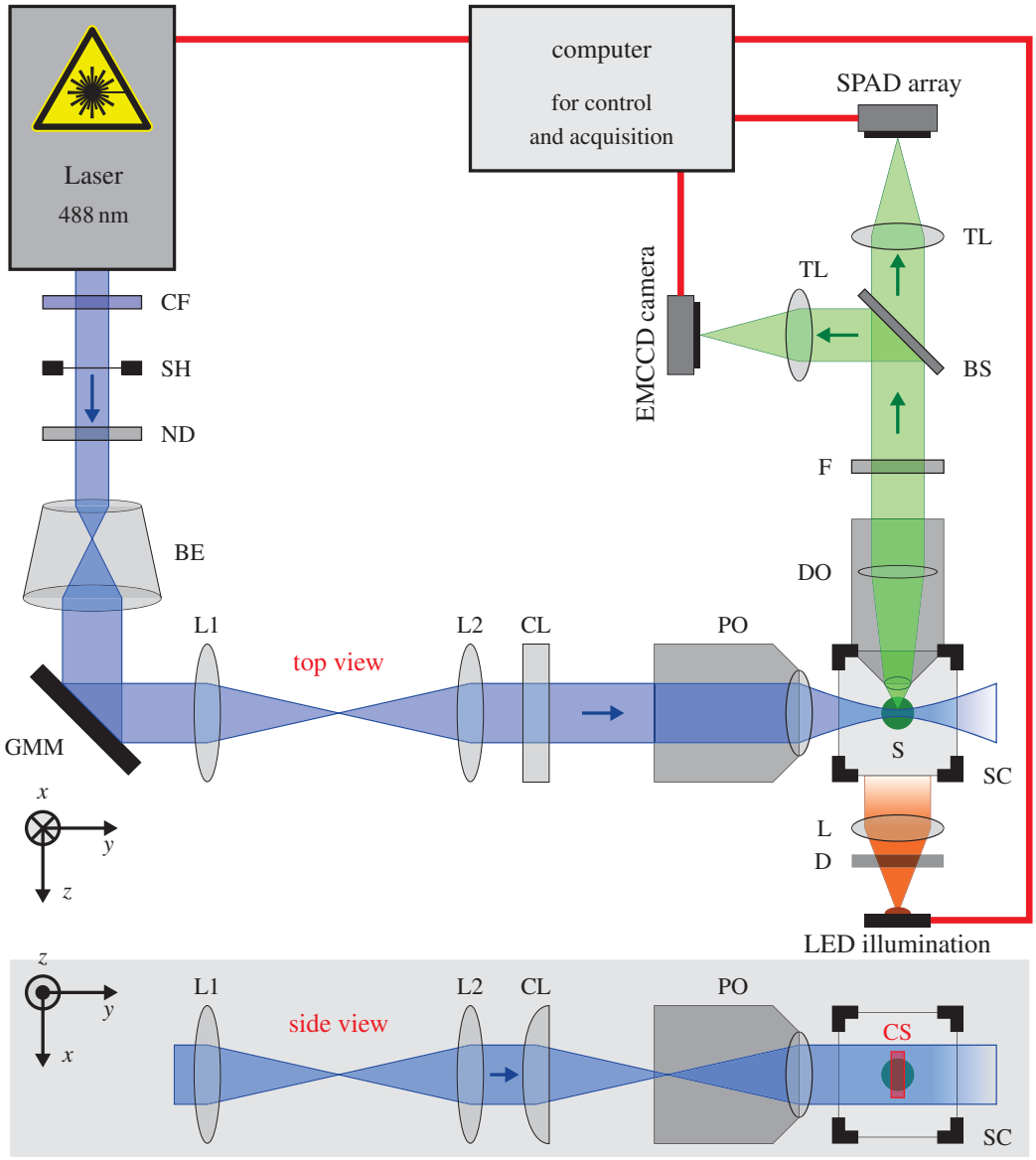
$$w_{\text{LS}}(y) \approx w_{\text{LS,min}} \cdot \sqrt{1 + \left(\frac{y}{1.58 \cdot d_{\text{LS}}}\right)^2}. \tag{2.18}$$

The parameters of the lightsheet depend on the sample medium. For water, the following values can be derived for the minimum thickness of the lightsheet w_{LS} and the longitudinal $1/e^2$-half width [119]:

$$\lambda = 488\,\text{nm}: \quad w_{\text{LS}} \geq 1.33 \quad \text{and} \quad d_{\text{LS}} \geq 20.2\,\mu\text{m}. \tag{2.19}$$

The geometry of the lightsheet is plotted in Figure 2.22a. Figure 2.22b shows a photograph of the lightsheet within the sample chamber.

*Additional laser light sources and dual color splitting optics that were not used in this work are not shown for simplicity.



BE	beam expander	DO	detection objective (60×, NA 1)	S	sample
BS	beam splitter	F	filter (longpass, transmission $\lambda > 488\,\mathrm{nm}$)	SC	sample chamber
CF	laser clean-up filter	GMM	gimbal mounted mirror	SH	shutter
CL	cylindrical lens	L...	lens	TL	tube lens
CS	location of CHSPAD	ND	neutral density filter, OD-0.5		
D	ground glass diffuser	PO	projection objective (20×, NA 0.3)		

Figure 2.20.: Schematic drawing of the SPIM setup as used during the course of this thesis. The upper part of the image shows the top view of the SPIM setup. The lower, shaded part shows the side view (frontal) of the illumination beam path. The position of the CHSPAD sensor is shown as a red rectangle. Connections to the computer used to control the instrument are shown in red. More details on the setup and the used components are given in Ref. [119].

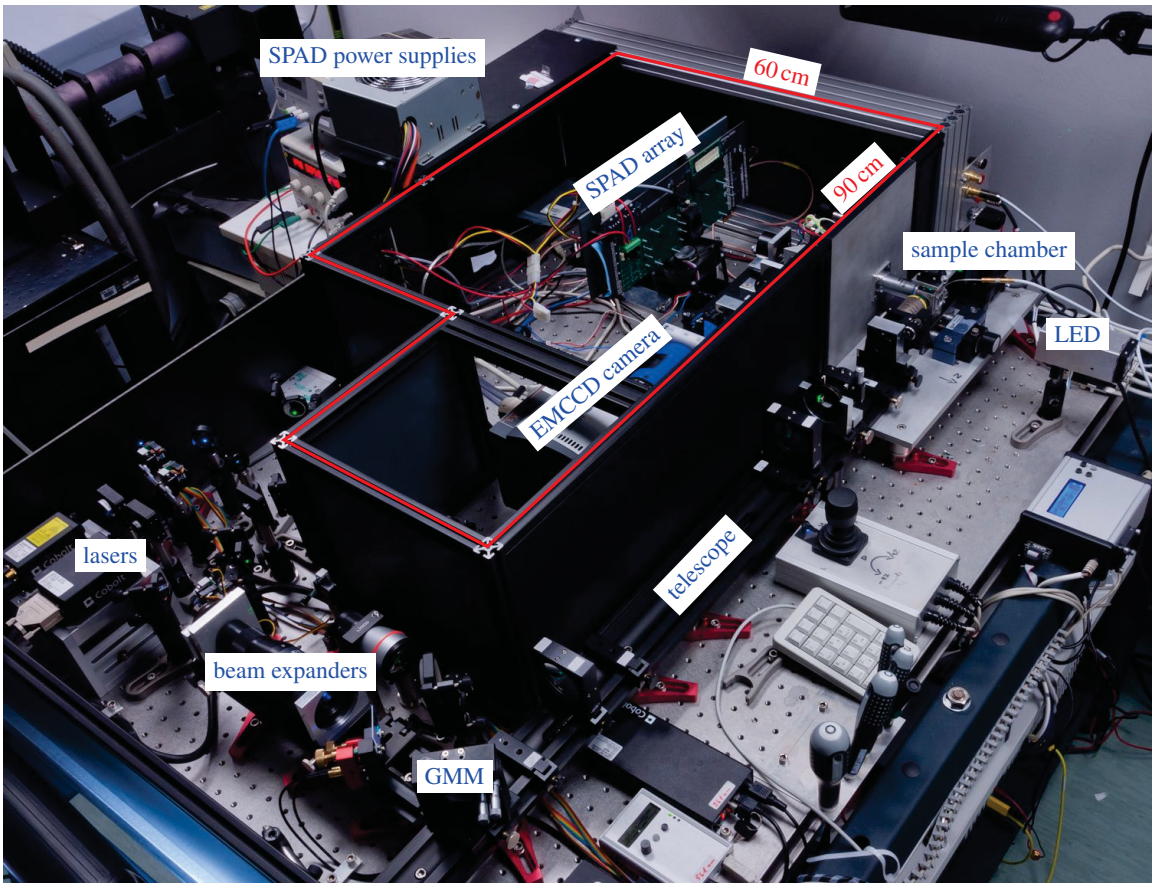


Figure 2.21.: Photograph of the SPIM setup. The red area of the light-proof box is covered during measurements.

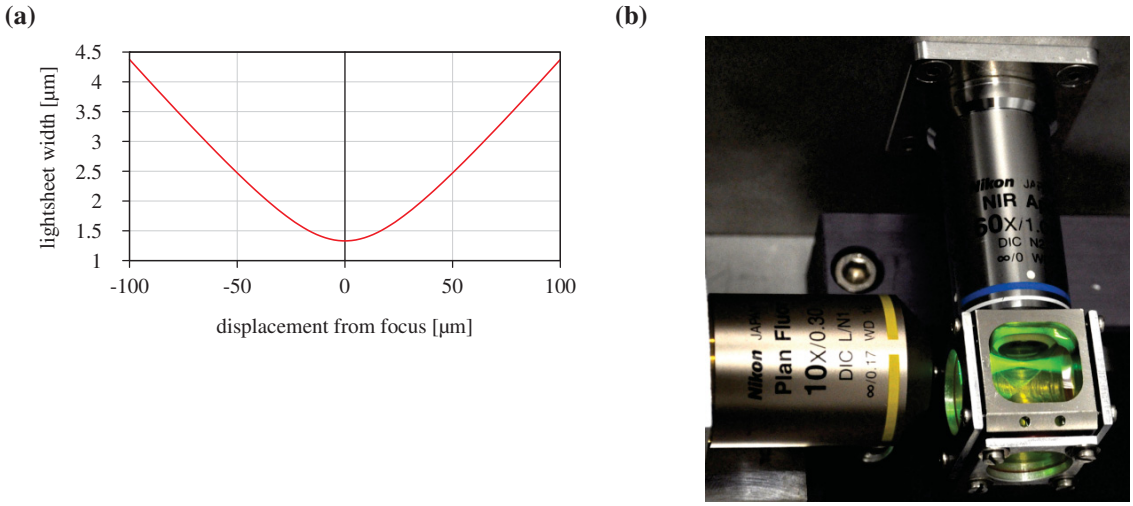


Figure 2.22.: Light-sheet in the SPIM setup. (a) Geometry of the lightsheet. Calculations were done for a 488 nm laser light source and a water filled sample chamber (SC). (b) Photograph of the lightsheet in the sample chamber. To visualize the lightsheet, a solution of *TetraSpeck* ∅ = 100 nm fluorescent beads was filled into the sample chamber. The projection objective is located in the bottom left, the detection objective on the top right.

sensor	type	resolution	pixel size d^2_{pixel} [µm^2]	d^2_{pixel} at 60× [nm^2]	FOV at 60× [µm^2]
ANDOR *iXon X3 860*	EMCCD	128 × 128	24 × 24	400 × 400	51.2 × 51.2
RADHARD2	SPAD	32 × 32	30 × 30	500 × 500	16.0 × 16.0
CHSPAD	SPAD	512 × 128	24 × 24	400 × 400	204.8 × 51.2

Table 2.6.: Geometrical parameters of the image sensors used in the SPIM setup. The two last columns give the respective pixel size and the field of view (FOV) in the image plane at 60× magnification. Data taken from Refs. [8, 33, 36].

The detection system of the SPIM setup consist of a water dipping objective (DO) with a 60× magnification (NA = 1.0, focal length $f_{\text{DO}} = 3.33$ mm). It is used to collect the fluorescence from the sample (S) excited by the lightsheet.

The detection objective is infinity corrected and a tube lens (TL) is used to focus down the light onto the detectors. A 50:50 beam-splitter (BS) or a mirror can be used to select one of the detectors or to use both in parallel. Different filters can be selected in a filter-wheel, which is placed between the detection objective and the beam-splitter. Typically, filters are used to select a certain range of the spectrum for detection (*i.e.*, fluorescence) and to block scattered light from the sample.

Table 2.6 shows the geometrical parameters for the three detectors that were used in this thesis work. Except for the *RADHARD2*, those match the size of a typical sample, *i.e.*, cell.

As the detection path comprises a water dipping objective, the sample chamber (SC) needs to be filled with pure water or buffer solution, depending on the sample. The sample, mounted on a three-axis translational stage, is brought in hanging from above. Liquid samples such a free dyes, are filled into small seal-able bags with the same refractive index as water (FEP foil). Cells are typically grown on glass cover slips and are directly hung into the buffer solution. Another possibility is to embed a sample into a gel, which in turn can then be inserted into the filled sample chamber.

2.3.2. Bleach correction

Fluorescent dyes, especially fluorescent proteins, have the tendency to loose their ability to emit photons over time (see section 2.2.2). This effect cannot be integrated into the fit models, thus requiring a cor-

rection of the raw dataset. In Ref. [176] a correction formula for a time-trace of intensity values $I(t)$ was proposed:

$$I^{(c)}(t) = \frac{I(t)}{\sqrt{f(t)/f(0)}} + f(0) \cdot \left(1 - \sqrt{f(t)/f(0)}\right), \tag{2.20}$$

where $I^{(c)}(t)$ is the corrected time-trace and $f(t)$ is a function which describes the progression of the fluorescence intensity $I(t)$. In Ref. [119] an extended mono-exponential model was successfully used:

$$f(t) = f_0 \cdot \exp\left(-\frac{1}{\tau_b} \sum_{i=1}^{N_f} f_i \cdot t^i\right), \tag{2.21}$$

which is also used in here. The parameters τ_b and f_i can be obtained from a fit of intensity time-trace. In this thesis, a polynomial degree of five was chosen ($N_f = 5$).

2.3.3. Experiment control and data evaluation software

The entire SPIM microscope, including lasers, laser shutters, the LED illumination, the EMCCD camera, the motorized stages to move the sample, etc., and even the *RADHARD2* and CHSPAD sensors are controlled by the *QuickFit3* open source software. *QuickFit3* was developed by JAN W. KRIEGER [122]. Apart from the experiment control, the focus of *QuickFit3* is data evaluation for fluorescence microscopy, especially correlation analysis and curve fitting. In case of the *RADHARD2* SPAD array and the CHSPAD, plug-ins have been developed for reading raw data files as well as raw data from the FPGA-based correlator (see section 5.5). Additionally, a live-view was integrated. Due to the limited data rate of the *RADHARD2*, correlation analysis of the raw data can be done with *QuickFit3*, too. In case of the CHSPAD, *QuickFit3* can be used to remote-control the data acquisition software *ngsoft* (see section 4.2.3) and allows displaying a real-time view of the sensor. Correlation analysis is done with an external program based on the CPU correlator ('correlator_vc_chspad_raw'). This software does the auto- and cross-correlation analysis. The outcome can be directly imported to *QuickFit3*. Furthermore, 'correlator_vc_chspad_raw' can be used for bleach correction (see section 2.3.2).

2.3.4. Characterization of the molecular detection efficiency

After the alignment of the microscope (see [119] for details), the molecular detection efficiency (MDE) of the setup is measured. The MDE is essentially the convolution of the point spread function (PSF) of the microscope and a function describing the shape of a single pixel [201, 241]. To measure the MDE, a gel cylinder that contains a small amount of *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads is scanned along the z-axis in steps of 200 nm [119]. The image stack is automatically evaluated using the *Beadscan Evaluation: PSF* plug-in in *QuickFit3*. In a first step bead candidates are detected by selecting the brightest spots per frame which are then merged with neighbors in z direction. In a second step, the bead candidates are fitted using a three-dimensional GAUSSIAN model yielding 6 parameters (position in the stack, and parameters of the GAUSSIAN volume). For later analysis, the largest width of the GAUSSIAN volume is taken as w_z, and the mean value of the shorter ones as w_{xy}. The median values of the largest and the smallest widths of the major axis of the GAUSSIAN fits were used as w_{xy} and w_z, respectively. To account for the inhomogeneous illumination of the CHSPAD with microlenses, the detection algorithm for bead candidates was adapted to apply a de-trending operation first (see section 6.3.5 for details on the de-trending method).

2.4. Measuring diffusion: fluorescence fluctuation fechniques

This section gives an introduction to the principles of fluorescence correlation spectroscopy. Starting with the basic concepts of correlation analysis (section 2.4.1), different algorithms to calculate correlation

functions are introduced (section 2.4.2). Using this as a starting point, the implementation of such an algorithm on different hardware architectures is described (section 2.4.3). Parts of this work have been already published as Ref. [155].

2.4.1. Signal Correlation

As already introduced in section 1.1.2, the method of choice in this thesis to assess the mobility of molecules was fluorescence correlation spectroscopy (FCS). This technique was pioneered by [142] [142–144] in 1974. Figure 2.23 illustrates the principle: A fluorescent label is excited and detected in a small sub-volume of the sample (some μm^3, (a)). In that volume, the particle number in time $N(t)$ is low enough to distinguish between single particles entering and leaving the volume. The measured fluorescence intensity $I(t)$ is proportional to the number of fluorescent molecules within the volume ((b)). Due to the continual motion, *e.g.*, diffusion, $N(t)$ is permanently fluctuating around its mean value $\langle N \rangle$, giving:

$$N(t) = \langle N \rangle + \delta N(t) \quad \Rightarrow \quad I(t) = \langle I \rangle + \delta I(t) \quad \text{with} \quad \langle \delta N(t) \rangle = \langle \delta I(t) \rangle = 0 \tag{2.22}$$

The fluctuations of the particle number and the detected fluorescence intensity are represented by $\delta N(t)$ and $\delta I(t)$. This is directly related to the speed of the particle (diffusion coefficient) and can be quantified using an autocorrelation analysis.

By means of an autocorrelation analysis the characteristic diffusion time is extracted ((c)). The normalized FCS autocorrelation function is defined as

$$G(\tau) = \frac{\langle \delta I(t) \cdot \delta I(t+\tau) \rangle}{\langle I(t) \rangle^2} = \frac{\langle I(t) \cdot I(t+\tau) \rangle}{\langle I(t) \rangle^2} - 1, \quad \tau > 0, \tag{2.23}$$

with the averaging function $\langle \cdot \rangle$ over an interval $[0...T]$:

$$\langle I(t) \rangle = \lim_{T \to \infty} \frac{1}{T} \int_0^T I(t)\, \mathrm{d}t. \tag{2.24}$$

Generally, a correlation function measures the similarity of two signals $I_A(t)$ and $I_B(t+\tau)$ shifted in time. In the case of an autocorrelation function, the self-similarity of the fluorescence intensity signal

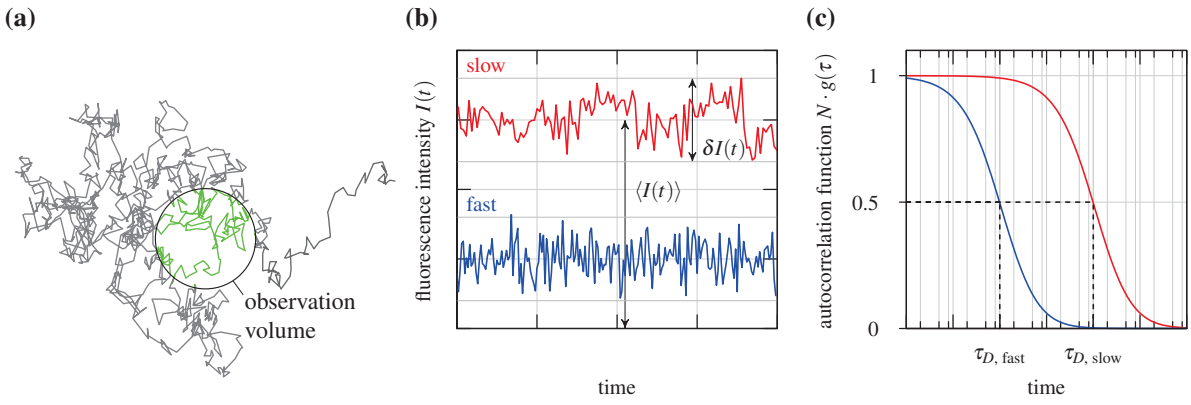


Figure 2.23.: Illustration of the principle of fluorescence correlation spectroscopy (FCS). (a) Sample with moving particles. Within the observation volume (black circle), the fluorescence is shown in green. (b) Fluorescence intensity as measured from two different samples, showing the fluctuations $\delta I(t)$ around the mean intensity $\langle I(t) \rangle$. A slow diffusing species is shown in red, a fast species in blue. (c) Autocorrelation curves calculated from the intensity traces in (b). The diffusion times $\tau_{D,\,\mathrm{fast}}$ and $\tau_{D,\,\mathrm{slow}}$ are defined as $g(\tau_D) = g(0)/2$.

$I(t)$ to its time-shifted variant $I(t+\tau)$ is measured. For a completely random intensity signal $I(t)$ (*i.e.*, white noise), the correlation function equals DIRAC'S function, $G(\tau) \propto \delta(\tau)$, and decays for all $\tau > 0$. When $I(t)$ contains a non-random component, the correlation $G(\tau)$ will be non-zero for a certain range of lag-times τ.

The first channel of the correlation function $G(0)$ (see equation (2.23)) is proportional to the average particle number in the observation volume:

$$G(0) = \frac{\left\langle (\delta I(t))^2 \right\rangle}{\langle I(t) \rangle^2} \propto \frac{\left\langle (\delta N(t))^2 \right\rangle}{\langle N(t) \rangle^2} = \frac{1}{\langle N \rangle}. \tag{2.25}$$

The last step is valid as the particle number $N(t)$ obeys POISSONIAN statistics:

$$\left\langle (\delta N(t))^2 \right\rangle \equiv \mathrm{Var}(N(t)) = \langle N \rangle . \tag{2.26}$$

Further, the correlation function decays for large τ:

$$G(\infty) = 0. \tag{2.27}$$

Discretization

A typical camera discretizes the continuous stream of fluorescence intensity $I(t)$ with a fixed time base, the frame time or integration time τ_{min}, into a finite sequence of N_T intensity measurements, with $N_T \cdot \tau_{\mathrm{min}} = T$:

$$I_n = \int_0^{\tau_{\mathrm{min}}} I\left(n \cdot \tau_{\mathrm{min}} + t\right) \mathrm{d}t, \quad n = 0, 1, ...N_T - 1. \tag{2.28}$$

THat givel, the continuous averaging function, equation (2.24), turns into

$$\langle I \rangle = \frac{1}{N_T} \sum_{t=0}^{N_T-1} I. \tag{2.29}$$

Autocorrelation

The discrete counterpart of the correlation function (equation (2.23)) is called correlation curve g. Care has to be taken not to bias the normalization $1/\langle I \rangle^2$. A viable choice is the "symmetric normalization" introduced by Schätzel et al. [190] and [189]:

$$g_{\mathrm{sym}}(\tau_k) = \frac{\frac{1}{N_T-k} \cdot \overbrace{\sum_{n=k}^{N_T-1} I_n \cdot I_{n-k}}^{=:g_k}}{\left[\frac{1}{N_T} \cdot \underbrace{\sum_{n=0}^{N_T-1} I_n}_{=:M^u}\right] \cdot \left[\frac{1}{N_T-k} \cdot \underbrace{\sum_{n=k}^{N_T-1} I_{n-k}}_{=:M^d}\right]} = g_k \cdot \frac{N_T}{M^u \cdot M^d}. \tag{2.30}$$

In the formula shown above, the continuous τ is also replaced by its discrete counterpart τ_k:

$$\tau_k = k \cdot \tau_{\mathrm{min}}, \quad k = 0, 1, ...N_T - 1, \tag{2.31}$$

The so called 'lag time'. Note that $M^u = g_0$ for a binary input signal, so the first correlation channel contains important information for the normalization.

Schätzel [189] showed, that a naïve implementation

$$g_{\text{asym}}(\tau_k) = \frac{\frac{1}{N_T-k} \cdot \sum_{n=k}^{N_T-1} I_n \cdot I_{n-k}}{\left[\frac{1}{N_T} \cdot \sum_{n=0}^{N_T-1} I_n\right]^2} \tag{2.32}$$

leads to additional noise as it includes contributions due to boundary terms [190]. A comparative study of different normalizations can be found in Ref. [52].

Cross-correlation

The formulas described above only measure the self-similarity of a signal. To be able to compare neighboring pixels, *e.g.*, to evaluate flows, equation (2.30) can be modified to its generalized form

$$g^{\text{CCF}}_{\text{sym}}(\tau_k) = \frac{\frac{1}{N_T-k} \cdot \sum_{n=k}^{N_T-1} {I^u}_n \cdot {I^d}_{n-k}}{\left[\frac{1}{N_T} \cdot \sum_{n=0}^{N_T-1} {I^u}_n\right] \cdot \left[\frac{1}{N_T-k} \cdot \sum_{n=k}^{N_T-1} {I^d}_{n-k}\right]}. \tag{2.33}$$

The values ${I^u}_n$ and ${I^d}_n$ represent discrete intensity values of two different pixels u and d (see below).

2.4.2. Algorithms

To calculate the correlation curve $g(\tau)$ several algorithms can be used. This section gives an overview of representative types of correlators. It follows the arguments given in Ref. [113] which also gives an analysis of the errors of the different correlators. The actual implementations of a correlator may vary depending on the exact input data type. Typically, the input stage (the so called 'front end') of the correlator differs and specific optimization can be taken.

A special position is taken by time-correlated single-photon counting (TCSPC) data. Instead of a continuous stream of accumulated or binary values (photon or no photon), such a detector provides photon arrival times. In case of low count rates, this can be thought of as kind of compression. Details on the algorithms can be found in Ref. [227].

Direct correlation

The simplest approach to calculate the (auto-)correlation function is to evaluate the correlation function directly. Therefore, the full sequence $\{I_n\}_{n=0\ldots N_T-1}$ is required and a correlator (typically a software implementation) evaluates equation (2.30) for an arbitrary (also logarithmically) spaced set of lag times τ_k. This gives an unbiased estimation of the autocorrelation function (ACF) ("direct correlation"). The computational cost for calculating equation (2.30) is of the order of

$$\mathcal{O}\Big(\underbrace{N_T}_{\text{Number of samples}} \cdot \underbrace{N_T}_{\text{Number of } \tau \text{ values}}\Big) = \mathcal{O}\left(N_T^2\right). \tag{2.34}$$

As the correlation function typically ranges from µs to s, a logarithmic spacing of τ_k is more computational cost efficient:

$$\mathcal{O}\Big(\underbrace{N_T}_{\text{Number of samples}} \cdot \underbrace{\log(N_T)}_{\text{Number of } \tau \text{ values}}\Big) \in \mathcal{O}\left(N_T \log N_T\right). \tag{2.35}$$

By choosing a certain number of τ_k, the complexity can be reduced to $\mathcal{O}\left(N_T\right)$. A major drawback of this algorithm is, that the entire dataset has to be held in memory during calculation. The implementations described in the following overcome this issue.

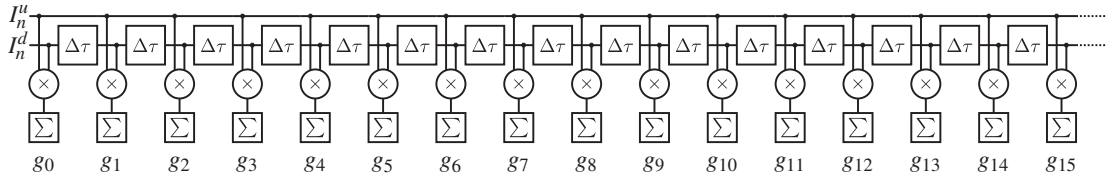


Figure 2.24.: Linear-τ correlator. The basic building block of the correlator is a shift register.

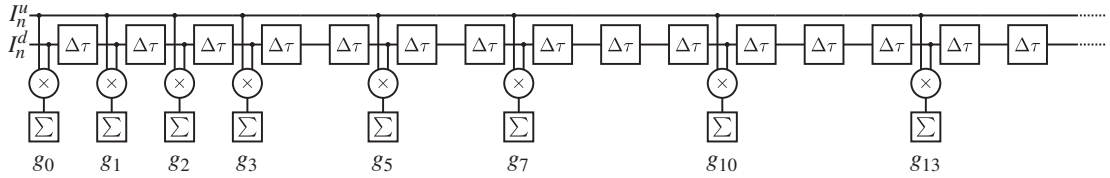


Figure 2.25.: Exponential-τ correlator. A linear correlator with an exponentially increasing number of shift registers in between.

Linear-τ correlator

Figure 2.24 shows a sketch of a so called linear-τ correlator [100], with linearly spaced lags:

$$\tau_k = k \cdot \tau_{\min}. \tag{2.36}$$

It is a direct implementation of g_k in equation (2.30), forming a shift register of individual lags (delay element and multiply accumulate operation) with a corresponding lag time τ_k. Due to the design of a correlator (see below), typically one input signal is delayed (d), whereas the other signal (u) passes the correlator without delay.

A major advantage of such a linear correlator over the direct evaluation of the correlation function is that input values can be discarded once they have passed the correlator. To be able to normalize the correlation function when a measurement is done, the monitor channels M^u and M^d have to be calculated additionally. If τ should embrace multiple decades in time, the consumption of hardware or computing resources is enormous. This requires for different approaches, which are discussed in the following.

Exponential-τ correlator

The exponential-τ correlator addresses the the high amount of hardware or processing power that is necessary if a linear correlator is used. To cover a larger range of lag times τ_k, additional shift registers are inserted between full featured lags. By increasing their length exponentially, a logarithmic scale is introduced. Figure 2.25 shows an implementation that uses the following relation:

$$\tau_k = \left(\sum_{i=1}^{k} 2^{\lfloor |i-3|/2+0.5 \rfloor} - 1 \right) \cdot \tau_{\min} \propto \left(2^{\frac{1}{2}} \right)^{k+1} \cdot \tau_{\min}. \tag{2.37}$$

The last relation arises as a result of a geometric series*.

Multiple-τ correlators

The latest result of correlator development are multiple-τ correlators, which are able to span decades of delay values τ_k with only a couple of hundred lags. Their major advantage over the exponential-τ correlator is, that by introducing an averaging of the intensity values I_n at some point, they can reduce hardware or computational effort tremendously. Such an averaging step, also called binning with a ratio

* $\lfloor x \rfloor := \max\{k \in \mathbb{Z} \mid k \leq x\}$

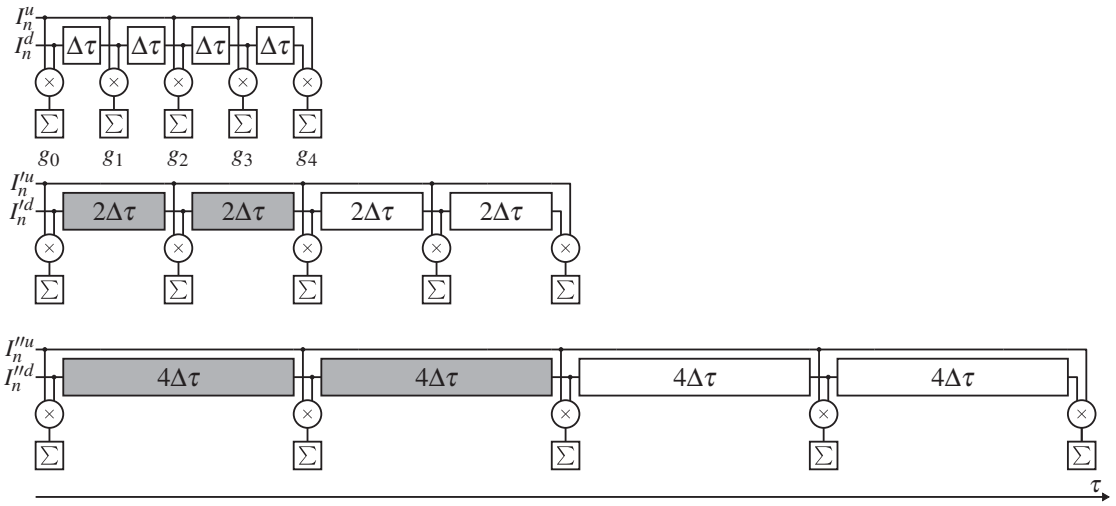


Figure 2.26.: Parallel multiple-τ correlator. Basic building blocks are linear correlators which are run in parallel. In each linear correlator the delay time is doubled ($m = 2$) and an averaging is done on the input signals. Due to the overlap (shown in gray), the first half of each linear correlator is redundant.

of m single values, is a convolution of the intensity values and a rectangular filter kernel. Equation (2.23) can then be written as

$$\begin{aligned} G'(\tau) &= \frac{\langle I(t) * h_{\text{rect}}(t; m \cdot \tau_{\text{min}}) \cdot I(t+\tau) * h_{\text{rect}}(t+\tau; m \cdot \tau_{\text{min}}) \rangle}{\langle I(t) * h_{\text{rect}}(t; m \cdot \tau_{\text{min}}) \rangle^2} - 1 \\ &= G(\tau) * h_{\text{triangle}}(t; m \cdot \tau_{\text{min}}). \end{aligned} \tag{2.38}$$

The kernel functions are defined as follows:

$$h_{\text{rect}}(t;T) = \frac{1}{T} \begin{cases} 1 & |t| < T/2 \\ 0 & \text{else} \end{cases} \tag{2.39}$$

$$h_{\text{triangle}}(t;T) = h_{\text{rect}}(t;T) * h_{\text{rect}}(t;T) = \frac{1}{T} \begin{cases} 1-|t| & |t| < T \\ 0 & \text{else.} \end{cases} \tag{2.40}$$

Parallel multiple-τ correlator. The parallel multiple-τ correlator described by Magatti and Ferri [140] is based on linear-τ correlators but makes use of binning to save computation time. Figure 2.26 shows a sketch of such a correlator. It is based on a set of S linear correlators ($s = 0, 1..., S-1$) having an increasing integration time and a binning ratio m:

$$\tau_{\text{min},s} = \cdot m^s \cdot \tau_{\text{min}}, \quad s = 0, 1, 2..., S-1. \tag{2.41}$$

For each linear correlator the delay values are given as

$$\tau_{k,s} = k \cdot \tau_{\text{min},s}. \tag{2.42}$$

A major drawback of such a design is that per linear correlator $\frac{1}{m}$ of each correlator is overlapping with the previous one. Commonly, a binning ratio of $m = 2$ is chosen for correlator implementation.

Serial multiple-τ correlator. The working principle of a (serial) multiple-τ correlator used in here was proposed by Schätzel [186]. Figure 2.27 shows a sketch of the design. It consists of B linear-τ correlators, so called blocks, typically comprising $L = 8$ or $L = 16$ single lags. Each lag is made up of a delay element

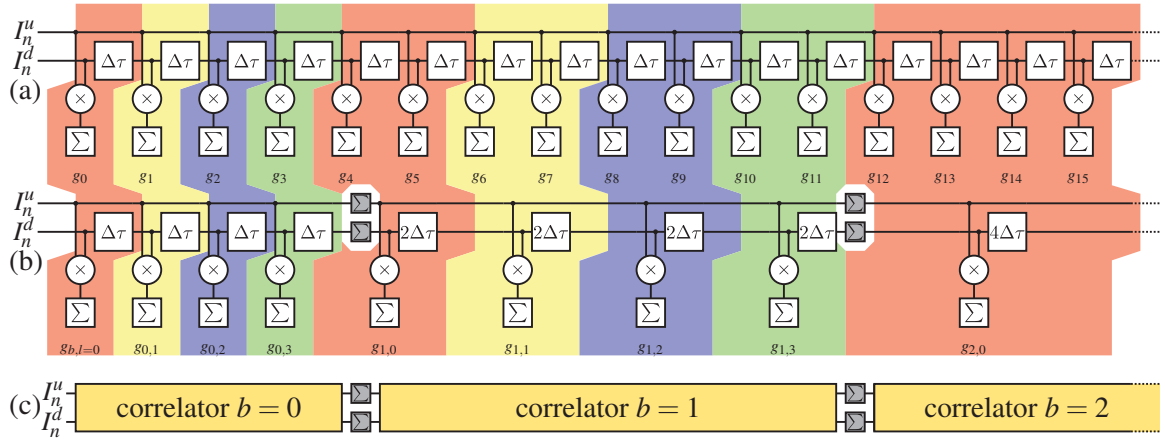


Figure 2.27.: Schematic drawing of a multiple-τ correlator. (a) shows a linear-τ correlator for comparison. (b) multiple-τ correlator consisting of blocks of linear-tau correlators with four lags each. Between adjacent blocks two consecutive global or local values are summed up. Corresponding channels are grouped with the same color. Global/un-delayed ($I^u{}_n$) and local/delayed ($I^d{}_n$) inputs are located on the left. For autocorrelation, the delayed and un-delayed signal inputs to the 0-th block ($I^u{}_n = I^d{}_n = I_n$) are identical. The $\boxed{\Delta\tau}$ blocks represent delay elements, the $\otimes$-blocks multipliers and the $\boxed{\Sigma}$-blocks accumulators. Typically $\Delta\tau = \tau_{\min}$. Panel (c) shows a schematic view of the multiple-τ correlator, where the linear-τ correlator building blocks are summarized by a single block.

($\tau_{\min}$), a multiplier ($\times$) and an accumulator ($\sum$). Usually, the minimum delay value is chosen to be the minimum integration time of the sensor ($\Delta\tau = \tau_{\min}$). In between adjacent linear blocks, m consecutive values of the global and the local inputs are summed up. The delay values of the following block are scaled in the same manner ($\tau_b = m \cdot \tau_{b-1} = m^b \cdot \tau_{\min}$). This introduces a semilogarithmic spacing and several orders of magnitude can be covered with ~ 100 lags. Each of the linear correlators estimates the correlation function at L linearly spaced lags. The lag times are given by the following relations:

$$\begin{aligned}
\tau_{0,0} &= 0 \\
\tau_{b,0} &= \left(\tau_{b-1,L-1} + m^{s-1}\right) \tau_{\min} \\
\tau_{b,l} &= \left(\tau_{b,l-1} + m^{s}\right) \tau_{\min} = \underbrace{\left(\sum_{i=1}^{b\cdot L+l} m^{\left\lfloor \frac{i-1}{L} \right\rfloor}\right)}_{k_{b,l}} \tau_{\min},
\end{aligned} \tag{2.43}$$

with $b = 0 \ldots B-1$ and $l = 0 \ldots L-1$.

The input samples for each block $I_{b,n}$ (n is the same index as in equation (2.30), so that $I_{b=0,n} = I_n$) are calculated as a summation of input values of the previous block, giving

$$I_{b,n} = \sum_{k=1}^{m^s} I_{b-1,n-k}, \quad \text{for } b > 0. \tag{2.44}$$

As shown in Ref. [113], the estimator $g_{\text{sym,multiple-}\tau}(\tau_{b,l})$ for a multiple-τ correlator equals the ideal correlation function $g_{\text{sym}}(\tau_{b,l} \cdot \tau_{\min})$ (see equation (2.30)) convolved with a triangular kernel (see equation (2.40)) of width m^b:

$$g_{\text{sym,multiple-}\tau}(\tau_{b,l}) = g_{\text{sym}}(\tau_{\min}) * h_{\text{triangle}}(t; m^b \cdot \tau_{\min}), \tag{2.45}$$

From one block to the next, the delay time is scaled with m in the multiple-τ scheme, reducing the input data rate of block $b+1$ by a factor of two in comparison to block b. Additionally, the frequency of execution of each block is scaled by $\frac{1}{m}$.

Optimizations. Instead of building a cascade of linear blocks, a single linear block can be re-used for all. Under the assumption, that the execution time is equal, the complete runtime $T_{\text{multi-}\tau}$ of the entire multiple-τ correlator is only a small multiple of the runtime $T_{\text{lin-}\tau}$ of the first linear correlator block:

$$T_{\text{multi-}\tau} = \overbrace{1 \cdot T_{\text{lin-}\tau}}^{1^{\text{st}}\text{ lin. corr.}} + \overbrace{\frac{1}{m} \cdot T_{\text{lin-}\tau}}^{2^{\text{nd}}\text{ lin. corr.}} + \overbrace{\frac{1}{m^2} \cdot T_{\text{lin-}\tau}}^{3^{\text{rd}}\text{ lin. corr.}} + \cdots \leq \sum_{n=0}^{\infty} \frac{1}{m^n} \cdot T_{\text{lin-}\tau} = \frac{m}{m-1} \cdot T_{\text{lin-}\tau}. \tag{2.46}$$

The equation shown above was calculated by exploiting the geometric series. Usually $m = 2$ is chosen, resulting in

$$T_{\text{multi-}\tau, m=2} = 2 \cdot T_{\text{lin-}\tau}. \tag{2.47}$$

Such an implementation requires additional memory in between the blocks to hold the state of the delay elements. Furthermore, a scheduling algorithm is needed which determines the order of blocks to be processed.

If the underlying hardware offers sufficient performance, even the linear correlator block can be replaced by a single lag. This lag processes all lags of the linear correlator serially. The content of the delay registers is then stored in a dedicated memory. This so called 'virtual correlator structure' was proposed by Jakob et al. [101].

Normalization. In case of a multiple-τ correlator, the symmetric normalization (*cf.*, equation (2.30)) yields the following [48]:

$$g_{\text{sym,multiple-}\tau}(\tau_{b,l}) = \frac{g_{b,l}}{2^b} \cdot \frac{N_T}{M^u_{b,l} \cdot M^d_{b,l}} \tag{2.48}$$

Both values $M^u_{b,l}$ and $M^d_{b,l}$ are usually calculated on-the-fly in so called monitor channels that simply accumulate the values $I^u{}_n$, $I^d{}_n$ that enter a specific lag. To minimize the computational costs, only the first lag in a block is equipped with such monitor channels, and $M^{(d,u)}_{b,l}$ are considered constant for a linear lag, introducing a certain error. If it is guaranteed that all local values or their accumulated variants $I_{b,l}$ pass all lags (*cf.*, next section), $M^u_{b,l} = M^u$ for all lags.

Under the assumption that I_n is a stationary random process and that $T > \tau_{s,p}$, the per-block monitors $M^d_{s,p}$ can be estimated in the following manner:

$$M^d_{b,l} = M^u \cdot \frac{N_T - k_{b,l}}{N_T}. \tag{2.49}$$

So only the M^u monitor channel is necessary for normalization. For the autocorrelation function the monitor channel equals the first channel $M^u = g_0$, but in case of a cross-correlator, both input signal I^u and I^d have to be accumulated separately.

Figure 2.28 shows the effect of the estimation in equation (2.48) (magenta) in comparison to a direct estimation of the ACF using equation (2.30) (green) and a multiple-τ correlator with a monitor channel per lag (blue). Data in (a) and (b) were obtained by correlating the input signal

$$I(t) = 1 + \sin(2\pi t/(1.51 \cdot 10^{-4})), \tag{2.50}$$

for which the exact ACF is known to be

$$g^{(\text{theoretical})}(\tau) = 1 + \cos(2\pi\tau/(1.51 \cdot 10^{-4})), \tag{2.51}$$

time t and lags τ are unit free). The data in (c) was created by simulating a $T_{\text{sim}} = 1\,\text{s}$ long FCS experiment with one diffusing species*. The computation was performed with an FCS simulation software described in Refs. [119, 239].

*The diffusion coefficient was $D = 20\,\mu\text{m}^2/\text{s}$ (corresponding to an intermediately sized protein in water), the simulation time-step of the random walk, as well as the minimum lag time were $\Delta t_{\text{sim}} = \tau_{\text{min}} = 1\,\mu\text{s}$. There were around 1.2 particles in the effective observation volume $V_{\text{eff}} \approx 0.4\,\mu\text{m}^3$ on average.

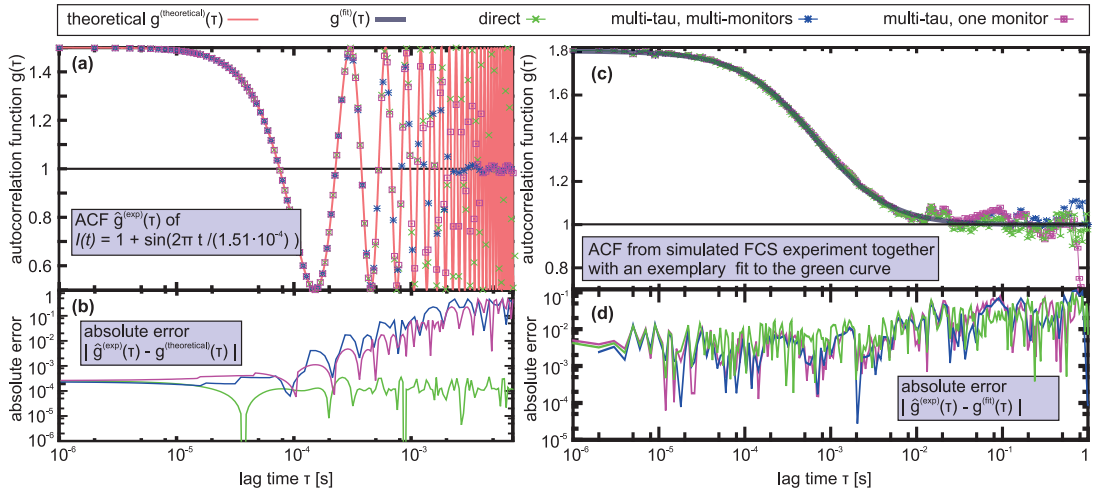


Figure 2.28.: Simulation results for different implementations of multiple-τ correlators. The left panels (a),(b) show simulations for a sine wave input signal $I(t) = 1 + \sin(2\pi t/(0.151\,\mathrm{ms}))$. The simulations on the right (c),(d) were created using an FCS simulation. The top graphs (a),(c) are estimates of the autocorrelation function using direct correlation from equation (2.53) (green), a multiple-τ correlator with one monitor channel per lag time (blue) and the estimated normalization from equation (2.48) (magenta). Graph (a) also shows the theoretical ACF $g^{(\mathrm{theoretical})}(\tau) = 1 + \cos(2\pi\tau/(0.151\,\mathrm{ms}))$ for the sine signal (light red). The lower graphs (b),(d) show the absolute deviation of the estimates from $g^{(\mathrm{theoretical})}(\tau)$ (b) and from a fit to the curves (d). The parameters resulting from the fit in (c) are the same within $< 2.5\%$ for all three estimates. Figure taken from Ref. [155].

Flushing the correlator. To ensure that all local values $I_{b,l}$ pass all lags, the correlator has to be run even after the actual measurement with $I_{n\geq N_T} = 0$. The last block ($b = B-1$) has to be run L times, the second last ($b = B-2$) has to be run $2 \cdot L$ times, the third last block ($b = B-3$) $4 \cdot L$ times and so forth. Thus, the number of values N_F that have to be fed into the correlator until all input values have propagated through all blocks is

$$N_F = 2^{B-1} \cdot L. \tag{2.52}$$

2.4.3. Theoretical FCS models

As the SPAD's diameters are small in comparison to the pixel pitch, they are treated as point like and the same model functions as in FCS were used. In contrast to that, the EMCCD camera with its rectangular pixels and an almost 100 % fill factor, different fit models were used (see [119] for details).

For evaluation, a theoretical model curve is fitted to the correlation curve that is obtained from the correlator. The model function can be established from the evaluation of the definition of the correlation function (see also equation (2.23)):

$$G(\tau) = \frac{\langle I(t) \cdot I(t+\tau) \rangle}{\langle I(t) \rangle^2} - 1, \quad \tau > 0. \tag{2.53}$$

The unknown intensity trace $I(t)$ can be modeled as the integral over the fluorescence of all particles:

$$I(t) = \phi \cdot \iiint \eta \cdot \varepsilon \cdot c(t,x,y,z) \cdot \mathrm{MDE}(x,y,z)\,\mathrm{d}x\,\mathrm{d}x\,\mathrm{d}z, \tag{2.54}$$

with the extinction coefficient ε, the quantum efficiency η (see section 2.2.3 for details on both), and an additional parameter ϕ, which models further loss in intensity due to the optics. The molecular detection efficiency (MDE) is used to describe the focal volume and $c(t,x,y,z)$ represents the particle concentration of the observed species. For a confocal microscope setup as well as for point like detectors in SPIM-FCS,

a three-dimensional, rotationally symmetric GAUSSIAN function with the $1/e^2$ width w_{xy} and $1/e^2$ height w_z is a good approximation:

$$\mathrm{MDE}(x,y,z) = \exp\left(-2\frac{x^2+y^2}{{w_{xy}}^2} - 2\frac{z^2}{{w_z}^2}\right). \tag{2.55}$$

Normal three-dimensional diffusion

By inserting equation (2.54) and equation (2.55) into equation (2.53), the FCS autocorrelation function for normal diffusion of a single component can be obtained [118, 142, 143]:

$$G_{\mathrm{norm}}(\tau) = \frac{1}{N}\cdot\left(1+\frac{4D_i\cdot\tau}{{w_{xy}}^2}\right)^{-1}\cdot\left(1+\frac{4D_i\cdot\tau}{{w_z}^2}\right)^{-1/2} \tag{2.56}$$

with the diffusion coefficient D and the particle number N.

The underlying BROWNIAN motion implies a relation of the diffusion time of (*cf.*, equation (1.9)):

$$\tau_D \propto \frac{\sqrt{V_{\mathrm{obs}3}}^2}{D}, \tag{2.57}$$

which becomes

$$\tau_D = \frac{{w_{xy}}^2}{4D}. \tag{2.58}$$

The parameters described above allow to calculate several derived values. Such as the effective volume

$$V_{\mathrm{eff}} = \pi^{\frac{3}{2}}\cdot w_{xy}^2\cdot w_z, \tag{2.59}$$

and the particle concentration in the focus

$$c = \frac{N}{V_{\mathrm{eff}}}. \tag{2.60}$$

Afterpulsing

Afterpulsing is a common artifact in single photon detectors, particularly in SPADs (see also section 2.1.1). It becomes visible as a fast decaying component superimposing the autocorrelation curve. As demonstrated in Ref. [119, p. 148], the afterpulsing that was prominent in the CHSPAD can be modeled using a single component power-law function:

$$G_{\mathrm{ap}}(\tau) = a_{\mathrm{ap}}\cdot\tau^{-b_{\mathrm{ap}}}, \tag{2.61}$$

with the amplitude a_{ap} and the exponent b_{ap}. It was also shown, that $b_{\mathrm{after}} = 1.1$ is an acceptable value for all pixels. An example curve of the afterpulsing of the CHSPAD is shown in section 4.2.4.

Normal three-dimensional diffusion for two foci

For two focus cross-correlation, equation (2.56) can be extended by a distance dependent term [27, 178]:

$$G_{2\mathrm{f}}(\tau) = \frac{1}{N}\cdot\left(1+\frac{4D_i\cdot\tau}{{w_{xy}}^2}\right)^{-1}\cdot\left(1+\frac{4D_i\cdot\tau}{{w_z}^2}\right)^{-1/2}\cdot\exp\left(-\frac{\delta_{xy}^2}{{w_{xy}}^2+4D\tau}\right), \tag{2.62}$$

where δ_{xy} is the lateral displacement of the two foci. Typically, no afterpulsing correction is applied, as it is not correlated for different pixels.

Oscillation fit model

Another model that was used in the course of this thesis is the oscillation fit model:

$$G_{\mathrm{osc}}(\tau) = A \cdot \cos(2\pi \cdot f \cdot \tau), \tag{2.63}$$

where A is the amplitude and f the frequency of the oscillation.

Although this model function is not related to diffusion, it was used for hardware and software tests. Typically, a frequency modulated light source was used to illuminate the sensor. As the excitation is precisely known, the entire chain of detection and evaluation can be tested.

Multi-component fit model and offset correction

To account for multiple species within the same observation volume, further components can be added to the fit model additively. If the correlation curve does not decay to zero as expected from the theory, an offset was added to the fit models.

2.4.4. Model fitting of correlation curves

The last step of the data evaluation process is the fitting of a model function $G(\tau)$ to the obtained raw correlation curves g_i. This was carried out using the *QuickFit3* software (see section 2.3.3) for each pixel i and all required auto- and cross-correlation curves.

For each correlation curve i, parameters are obtained by solving the least-squares optimization problem for all lag times τ:

$$S_i(\beta_i) = \underset{\beta_i}{\operatorname{argmin}} \sum_{\tau} [G(\tau, \beta_i) - g(\tau)]^2 \tag{2.64}$$

with the parameter vector

$$\beta_i = (D_i, N_i, w_{xy,i}, w_{z,i}, \dots), \tag{2.65}$$

for a three-dimensional diffusion model as described above. The vector β contains all required parameters of the model function. Typically, some of these parameters are fixed to a constant value, which was obtained differently.

Model fitting was done throughout this thesis work using a non-linear least-squares LEVENBERG-MARQUARD fit algorithm. As a software-implementation, *lmfit* was used [242]. Due to the sheer amount of pixels, an unattended fitting procedure is desired and it appeared that choosing optimal initial values is crucial for the fit algorithm to converge for a high number of pixels. Typically, N and D are used as free parameters, whereas w_{xy} and w_z are fixed (*e.g.*, obtained from bead-scan, see section 2.3.4). When more free parameters (except for the amplitude of the afterpulsing or a constant offset) are used, the fits tend to not converge any more.

Binning

The correlation curves of single pixels are typically noisy and fitting of a model function does not work well in many cases. When using imaging fluorescence correlation spectroscopy (imaging FCS) techniques, one can take advantage of the large number of pixels that measured simultaneously. A valid approach is then to bin raw intensity values of neighboring pixels. This increases the light intensity but also the focal volume of the resulting binned pixels. A major drawback is that the raw data has to be kept or the size of the binning region has to be chosen a priory to the measurement. Also, if a dedicated hardware correlator is used, this step might impose modifications of the hardware.

A different method which was used in the course of this thesis does not rely on the raw data but on the correlation curves of each pixel: Neighboring pixels are binned post-correlation by averaging correlation curves $g(x,y,\tau)$ of different pixels (x,y):

$$\bar{g}(x,y,\tau) = \frac{1}{b^2} \sum_{x_b=-b/2}^{b/2} \sum_{y_b=-b/2}^{b/2} g(x+x_b, y+y_b, \tau), \tag{2.66}$$

with b being the number of pixels to be used for binning. In principle, also a GAUSSIAN kernel can be used or the median value instead of this rectangular shaped kernel. This procedure is only valid under the assumption that changes in the diffusion coefficient are small in a local binned region and this step effectively reduces the resolution of imaging FCS.

2.5. Application accelerators

To calculate the correlation function of every single pixel of the CHSPAD in real time, high performance computing platforms are required. Starting with the general concepts of acceleration, the architectural properties of current CPU, graphics processing units (GPUs) and FPGAs are introduced.

2.5.1. Introduction

The power consumption of any digital circuit scales linearly with its clock frequency [85]. Thus, acceleration by increasing the clock frequency of a single CPU core found its end in the mid-2000s, when the so called "power ceiling" inhibited further speed increase as cooling became more and more impractical [166]. At that point, the focus started to shift towards parallelization and first chips with multiple processor cores entered the market. But also other architectures benefit from the still-increasing number of transistors: Similar to CPUs, also GPUs gain compute power by adding additional cores, or blocks of cores. FPGAs benefit from a large amount of logic resources by increasing the overall length of the processing pipeline, or by multiple instantiations of the same pipeline.

Limits of Acceleration

The maximum achievable speedup η_s due to parallelization can be assessed by AMDAHL'S law [7]:

$$\eta_{s,\mathrm{A}} = \frac{T}{t_s + t_{N_P} + \frac{t_p}{N_P}} \leq \frac{T}{t_s}, \tag{2.67}$$

with the runtimes of the parallel and serial code t_p and t_s, the total runtime $T = t_s + t_p$, the number of parallel processors N_P and the time required for additional synchronization t_{N_P}. Even for an infinite number of processors, the serial section cannot be accelerated any further, which leads to a fixed upper limit. For example, a program with only 5 % sequential code only gains a maximum speedup of 20.

If contrary to the assumption of AMDAHL, the size of the dataset is not fixed, but the execution time, the following speedup can be gained by increasing the problem's size (GUSTAFSON'S law) [79]:

$$\eta_{s,\mathrm{G}} = N_P + (1 - N_P) \cdot \frac{t_s}{T}. \tag{2.68}$$

A typical application in high performance computing (HPC) is real-time processing of data. Therein, the total amount of processing time is fixed. By using more compute elements in parallel, it is possible to calculate more datasets in parallel, so that GUSTAFSON'S law applies, and a linear speedup is expected.

Flinn's Taxonomy

Different computer architectures can be subdivided based on the instruction stream or data stream parallelism. Almost five decades ago, FLYNN described a taxonomy which is still in use today to describe hardware architectures:

- **single instruction, single data (SISD):** Classic single core computers, that perform their tasks serially, like PCs or workstations. The hardware is typically based on the VON-NEUMANN or HARVARD architecture. Acceleration is possible by reordering instructions and by pipelining their execution, which can be done in the background without interaction from the programmer. Data dependencies limit the total length of the pipeline and the overall performance.

- **single instruction, multiple data (SIMD):** Typical (classical) examples are mainframe computers and super-computers with array or vector processors. Fast execution of a single operation on multiple data streams (*e.g.*, used for image processing). Modern architectures typically feature SIMD instructions through instruction set extensions. Examples are the multimedia extension (MMX), the soft streaming extension (SSE) and the advanced vector extensions (AVX) [98]. A second representative of this class of computers are graphics processing units with thousands of compute units that perform the same instructions in parallel.

- **multiple instruction, single data (MISD):** Contrary to vector processors, where each pipeline stage is a fragment of a single instruction, this architecture can be considered as a pipeline of independent functional units operating on the same stream in parallel [67]. For example FPGA-based image processing pipelines with independent compute units fall into this category [55] and also dataflow computing [66]. Another representative are fault tolerant systems. Here multiple compute units perform the same operations on a single data stream and a final stage must agree on the results [205].

- **multiple instruction, multiple data (MIMD):** Typically, a MIMD machine is obtained by creating a set of independent SIMD or MISD processors that share a globally available memory.

Well established concepts for programming traditional and modern computers (SISD, SIMD and MIMD) like imperative, functional or logic computer languages do not fit the MISD architecture. Here a dataflow description is needed to build a pipeline of independent functional units. Until now, no full transformation is possible, that achieves maximum parallelism [194].

Single instruction, multiple thread (SIMT). A fifth class of architectures was recently added to FLYNN'S taxonomy: The SIMT architecture is comparable to the SIMD paradigm and was first introduced by NVIDIA as a parallel execution model for GPU platforms [131]. A set of SIMD processors virtually execute a much larger amount of threads in parallel by executing groups of threads serially. If a set of threads stalls, ideally another group of threads is ready to be executed and takes over the computing resources. This technique can be used to hide memory latencies and limits the instruction fetching overhead.

2.5.2. Dataflow computing

In dataflow computing, a program is modeled as a direct graph. Data flows between compute nodes along the arcs. If the underlying hardware is an FPGA, mapping is done by instantiating pipelines, and operations are performed in parallel on the data stream (MISD architecture). Classic programming concepts focusing on commands, in line with VAN-NEUMANN concepts, have to be substituted by inherently parallel approaches. This usually leads to a full redesign of the software.

Data is transferred between all nodes in a pipeline in each clock cycle. Therefore, an overall much higher bandwidth is achieved. This may help to reduce the impact of the processor-memory performance gap, which is the difference in growth in performance of CPUs and memory [85]. Thus, only the memory bandwidth for the sources and sinks are significant, whereas the overall lower clock speed reduces the requirements for the node-to-node connections. With all nodes working in parallel, the hardware is utilized evenly. A typical application of dataflow computing is image processing, and usually each node handles a single pixel's value.

2.5.3. Field programmable gate arrays

The terminology in the following section is based on the XILINX *documentation [243].*

A field programmable gate array (FPGA) is an integrated circuit (IC) whose function is configured post manufacturing by the customer. First FPGAs were introduced 1985 by XILINX, Inc.[23].

Usually, programmable logic is used when general-purpose hardware does not meet requirements and designing an application-specific integrated circuit (ASIC) is no option. Being configured within seconds, FPGAs can be chosen if the design maps onto the available resources, offering speed comparable to custom hardware. Using also the same descriptive language as used for ASIC design, porting existing designs is relatively easy.

Since their market launch in 1985, FPGAs have been used in various domains. Started as classical glue-logic, FPGAs now have evolved to ASIC replacements in low to medium range quantities. They are also used for ASIC prototyping or emulation. Due to their flexibility, FPGAs have ever since been used as domain specific co-processors and entered the market for HPC as application accelerators, for example in astronomy simulation[81].

In this work, FPGAs play an important role at different stages: They are used as ASIC replacements and prototypes for the control and the readout of the detectors as well as application accelerators for signal correlation analysis.

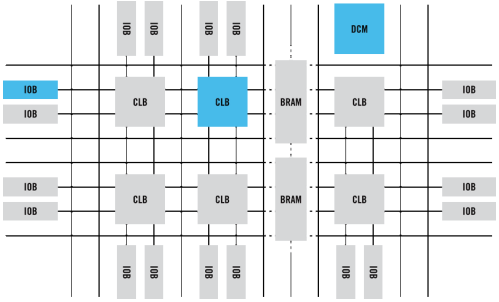
2.5.4. Circuitry

Figure 2.29a shows the basic internal structure of FPGAs that has not changed ever since: a large array of CLBs is embedded into a programmable network of wire interconnects, the so called 'fabric'. A configurable logic block (CLB) consists of lookup tables (LUTs) and registers (*i.e.*, flip-flops) that are grouped into slices (*cf.*, Figure 2.29b), allowing for implementation of either arbitrary combinatorial n-ary binary functions or sequential circuits (*cf.*, Figure 2.29c). Larger functions are realized by interconnecting multiple CLBs. Each CLB is connected to a switch matrix, which allows configuring the wire interconnects. Additional connections to adjacent blocks allow for fast carry chains, *etc*. Although the registers within the CLBs can be used as memory, too, dedicated memory blocks, so called block RAM (BRAM, see below), can be found in most FPGAs.

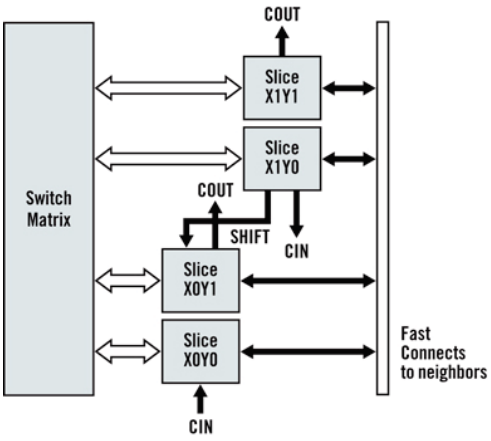
A slice found in a VIRTEX 6 device contains four LUTs featuring six inputs (red boxes of Figure 2.29c) and two outputs each. Eight single bit storage elements (green) that can be used, *e.g.*, as flip-flops (FFs), and various multiplexers. The combinatorial logic of a slice is generated by the six-input LUTs, which is realized as a small $2^6 \times 2$ static random access memory (SRAM). By utilizing the inputs as address lines, it is used to look-up the result of an arbitrary six-ary binary function. The truth table is programmed during the configuration of the FPGA.

Special functions, that cannot be represented as a circuit in the fabric, are realized as additional blocks within the FPGA. Examples are input/output blocks (IOBs), digital clock managers (DCMs), blocks related to the (re-) configuration process, multi-Gbit transceivers, *etc*. To save logic resources, commonly used logic is available as dedicated blocks, too. These blocks range from multipliers (*e.g.*, DSP cells, see below) through PCIe blocks to RISC-processors (PowerPC on VIRTEX-4 or ARM on VIRTEX-6,

(a) Simplified FPGA Block Structure



(b) Basic configurable logic block (CLB) structure



(c) Diagram of a single slice

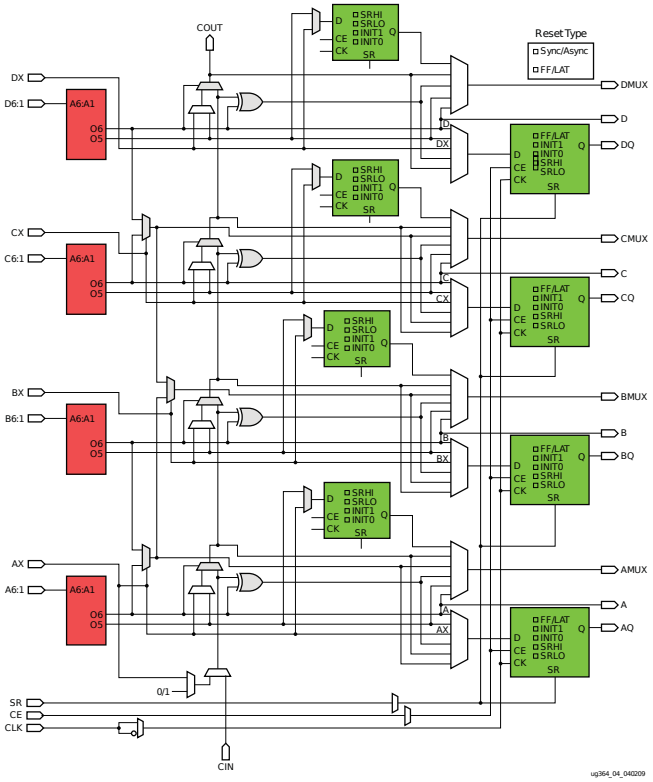


Figure 2.29.: Internal structure of an FPGA. Three different levels of abstraction are shown for a XILINX VIRTEX-6 FPGA. (a) Coarse grain structure of an FPGA. (b) Structure of a CLB. (c) Structure of a single slice. LUTs are shown in red; flip-flops in green. Two slices form a single CLB. Images are taken from Refs. [244, 253].

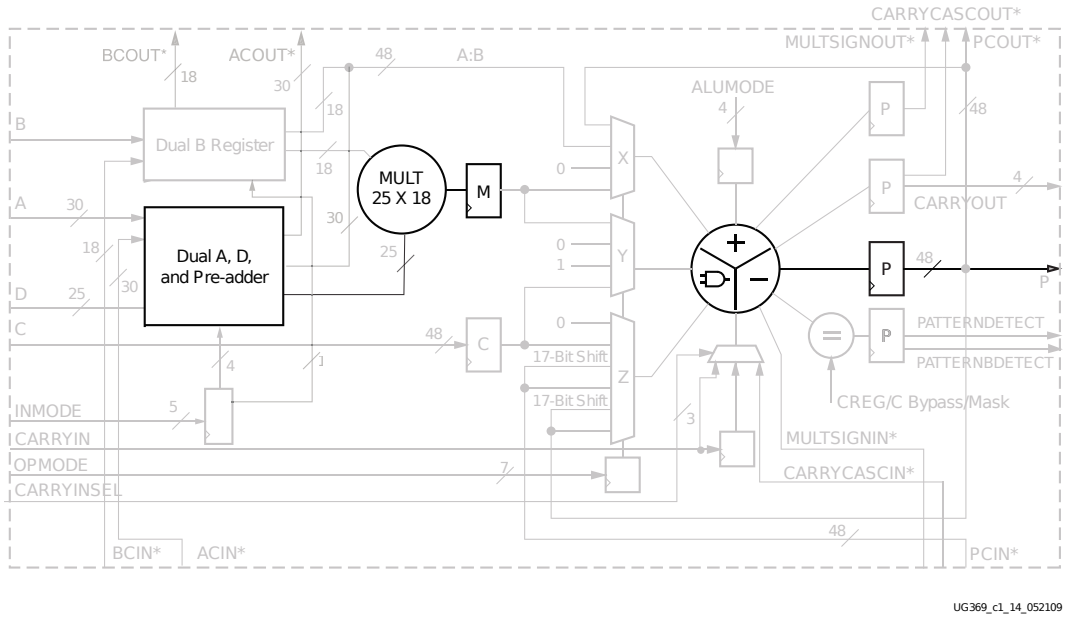


Figure 2.30.: Schematic drawing of the embedded functions in a *DSP48E1* slice as found in XILINX VIRTEX-6 FPGAs. The pre-adder stage (left), the 25×18 bit multiplier (center) and the ALU stage (right) are highlighted. Flip-flops are placed between the functional blocks to allow a high degree of pipelining. Adapted from Ref. [250].

[246, 252]) The additional layer of logic abstraction leads to a more then 30-fold overhead in silicon area consumption when the same logic function is implemented compared to an ASIC of the same CMOS process. By making heavy use of special function blocks, the overhead can be brought down below five [124].

Each component within the FPGA is configured with a few single bits. The total amount of bits is called the 'bitstream' and determines the full configuration of the device. This configuration has to be loaded each time the FPGA is re-powered. For a XILINX VIRTEX 6 device, the size of the bitstream is on the order of 10 MB [248].

2.5.5. Block RAM

Memory within the FPGA fabric can either be implemented using the slice registers or by using dedicated resources, the so called block RAM (BRAM). A BRAM is a dedicated two port memory containing several kilobits (VIRTEX-6: 4.5 kB per BRAM and up to 912 BRAMs per FPGA). Its flexible, configurable interface allows input data widths of 1 bit to 36 bit. Typically, the dedicated blocks have features beyond storing and loading data like error correction or first in - first out buffer (FIFO) logic [255]. Both ports, of which each can be used for reading or writing, of the BRAM are truly independent, and may even be located within different clock domains.

2.5.6. Digital signal processors

Most arithmetic functions based on generic integer or fixed point data types, *e.g.*, addition, subtraction, or logical operations, can be implemented in the fabric logic with a low resource usage and acceptable speed. However, when multiplication is required, a large amount of logic resources is necessary. The required resources scale with $\mathcal{O}\left(n^2\right)$ in the number of bits n for a fully pipelined multiplier (WALLACE-tree algorithm [230]).

As multiplication is often required in digital signal processors (DSPs), current FPGAs feature dedicated multipliers. The schematic of a DSP slice found in a XILINX VIRTEX-6 FPGA is shown in

Figure 2.30. Depending on its configuration, once per clock cycle the term $P = ((A \pm D) \times B) + C_{\text{in}}) \pm C$ can be evaluated. Adjacent DSP slices can be inter-connected and/or cascaded for wider operations. Applications are fast Fourier transforms, floating-point calculations and filter functions [250].

2.5.7. Hardware description languages

When first FPGAs entered the market in the mid-1980s, less than 100 CLBs were present in a single chip (*i.e.*, XILINX *XC2064*: 8×8 CLBs). At such a low level of complexity, designing the logic manually was still feasible. However, with an increasing amount of available resources in FPGAs and also in ASICs, software was required to ease the development and to allow describing the hardware on a higher level of abstraction.

Very High Speed Integrated Circuit Hardware Description Language

The Very High Speed Integrated Circuit Hardware Description Language (VHDL), among *Verilog*, is one of the two widely used hardware description languages (HDLs). It is used to describe digital systems system such as FPGAs and integrated circuits. VHDL was first standardized by IEEE in 1987. Its syntax is influenced by Pascal and Ada. In contrast to procedural programming languages such as *C* and assembly code which run sequentially, VHDL is a dataflow language which allows the description of a concurrent system. It is used to write text models to describe a logic circuit. This is done on the register transfer level (RTL), which is a design abstraction that models the flow of signals through combinational logic between registers. One of the key features of VHDL is that it allows modeling as well as simulating a circuit before its translation into real hardware. Thus, only a subset of the language itself can be synthesized into actual hardware. Higher level constructs, for example loops with a variable number of iterations, are limited to simulation only. Another key benefit is its portability: as the description is independent of the underlying hardware architecture, logic circuits that were created once can be ported between those easily.

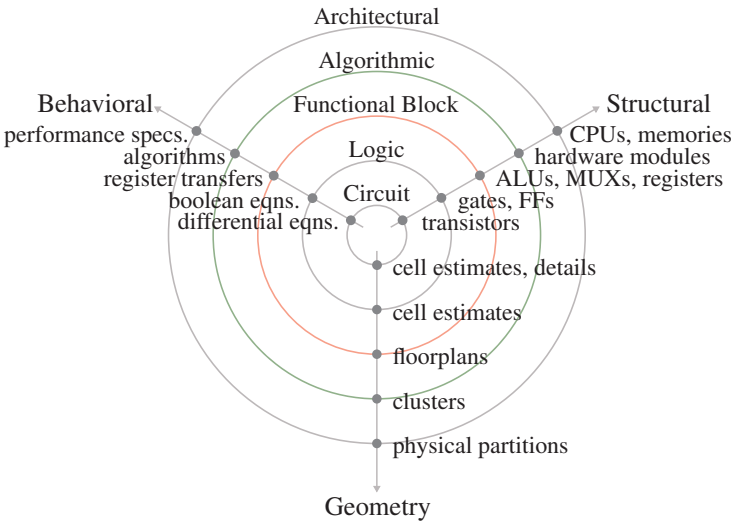


Figure 2.31.: WALKER-GIJSKI diagram showing the levels of abstraction versus domains of description. The RTL that is described with common HDLs (*e.g.*, VHDL and Verilog) is shown in red. A high-level hardware description language moves the description to the algorithmic layer, which is shown in green. Adapted from Ref. [229].

High-level hardware description

The HDLs described above provide a certain level of abstraction when compared to gate level design using a schematic editor. However, since the creation of HDLs, efforts were made to raise hardware description to a higher level.

Figure 2.31 shows a WALKER-GIJSKI diagram of the different levels of abstraction. Abstraction increases from the inner towards the outer rings. The actual undisclosed hardware of FPGAs itself is found at the center of the diagram (circuit layer). The layer above, the logic layer, contains the basic logic elements of an FPGA (*e.g.*, LUTs, FFs, *etc.*). VHDL and *Verilog* describe hardware on the layer of functional blocks, the RTL.

A higher level of abstraction significantly increases the productivity of a programmer and reduces the risk of bugs. Several frameworks were proposed for this purpose, which usually fall into five categories [14]: These frameworks are either based on existing HDLs as extension (System-*Verilog*) or *C*-based frameworks where only a subset of the language itself is allowed and additional concurrent structures are added (*i.e.*, *Handel-C*). A third category are CUDA/OpenCL based frameworks. Due to the popularity of GPU computing, which exploits the massive-parallel architecture of GPUs (SIMD), several CUDA/open computing language (OpenCL)-to-VHDL/*Verilog* were proposed. Another category are high-level language-based frameworks, which make use of highly abstract, object oriented, modern programming languages. Some of these are based on *Java* and typically a graph oriented description is used. The last category are Model-based frameworks. Examples are NATIONAL INSTRUMENTS *LabView* or MATHWORKS *Matlab* HDL Coder. These domain-specific frameworks are useful if the programmer is already familiar with the software but does not have further expertise in HDL programming.

2.5.8. FPGA design implementation flow

The conversion step from the RTL description of a circuit to the configuration file, that can be loaded into the FPGA ('bitfile'), is called design flow [251]. The required steps are done automatically by the tools provided by the FPGA vendor. Typically, the process is done in two steps:

Logic synthesis. After syntax checking, the source code is parsed and the recognition of specific language patterns such as state machines, memories, DSP cells, *etc*. is done. Additionally, a low level optimization is performed, which includes a resource sharing check. As a result, the synthesis process generates an architecture-specific 'netlist'. This netlist contains a representation of the design by describing it as a set of logical elements and their interconnections.

Implementation. The actual implementation of the netlist is done in four steps: As a first step, the *Translate* process converts incoming netlists to vendor specific primitives. Second, the *Map* process fits the design to the hardware resources of the target device. Then, the *Place and Route* process places and routes the design according to the predefined timing specifications. Lastly, a 'bitfile' is created from the results that can be downloaded to the target device.

2.6. Multi-core CPUs

As PCs are all around, a natural choice for application acceleration is a workstation computer with a conventional CPU. Thus, the correlation algorithm was implemented and evaluated on two different CPUs (INTEL *Haswell* and AMD *Piledriver*, see below). Table 2.7 shows performance related data of both CPUs.

Although being made for general purpose, todays (server-) CPUs offer up to almost 20 individual physical cores on a single chip (*e.g.*, INTEL *Haswell*-EP), additional virtual cores (*i.e.*, hyper-threading),

CPU	cores	256 bit FMA units per core	clock GHz	SP-FMA performance AVX [Gflops/s]	SP-FMA performance SSE [Gflops/s]	memory bandwidth GiB/s
INTEL *i5-2520M* (*Haswell*)	8	2	3.4	217 [64]	109 [32]	25.6
AMD *FX-8320* (*Piledriver*)	8	1	3.5	112 [32]	112 [32]	29.9

Table 2.7.: Theoretical peak performance of both used CPU platforms. The number of cores also counts the virtual cores due to hyper-threading. Values in squared brackets represent the performance per clock cycle.

and dedicated vector units. Current SIMD vector units perform the same operation on eight single precision floating-point values in parallel. If all of this parallelism is working together, almost 300 single precision floating-point operations can be performed in a single clock cycle. An exceptional advantage of using a conventional workstation computer as a platform is the portability. If the code is written in a generic form, is can be run on almost any computer. On the contrary, GPU or FPGA code is typically limited to a specific hardware platform or vendor.

2.6.1. SIMD instruction set extension

First SIMD features were introduced to CPUs as the multimedia extension (MMX), which allowed to concurrently process two 32 bit integers, or four 16 bit integers, or even eight 8 bit integers. In 1999, the soft streaming extension (SSE) was introduced with the INTEL *Pentium-3* to counter AMD's *three-dimensionalNow!* instruction set extension, which added floating-point support to the existing integer-only MMX. On current 64 bit computers, the SIMD register file comprises 16 independent 128 bit registers, that can be used as two 64 bit double precision floating-point values, four 32 bit single precision floating-point values or integers of the same size. SSE registers underwent a doubling in width with the AVX instruction set extension from 128 bit to 256 bit. Additionally, the three operand instruction format was introduced, to support non-destructive operations. Still, the 128 bit instructions are supported. With AVX 2, introduced with the INTEL *Haswell* microarchitecture, most integer commands were expanded to make use of the full 256 bit-wide AVX registers.

Although widely used compilers can make use of SIMD instructions, unattended vectorization of source code, so called auto-vectorization, is still limited. Here, the compiler needs to understand the program at a higher level. Thus, manual optimizations can improve performance significantly.

Recent SIMD instruction set extensions are supported by compilers via so called *intrinsics*. These instructions are abstractions for the corresponding assembler code. A comprehensive list of intrinsics for the INTEL-*C*-compiler, which is also applicable for most other *C*-compilers, can be found in Ref. [42].

Fused multiply accumulate

The fused multiply accumulate (FMA) operation is a variant of the multiply accumulate (MAC) operation on floating-point values, whereby rounding, if any, is done last:

$$a \leftarrow a + (b \cdot c). \tag{2.69}$$

This operation is available in INTEL and AMD microprocessors for single precision or double precision operands since 2011, either as a three-operand instruction (FMA-3) or a four-operand instruction (AMD only). Beside the higher precision, these operations are typically executed within a single cycle of the CPU, leading to a doubling in the total floating-point performance when used.

Further developments

AVX 3 (also known as AVX-512) extends the 256 bit wide SIMD registers to 512 bit. At the same time it doubles their amount. This new instruction set will be introduced with the upcoming INTEL *Skylake* microarchitecture and INTEL *Xeon Phi Knights Landing*.

2.6.2. INTEL *Haswell*

One of the two CPU platforms that were used in the course of this thesis was the INTEL *i7-4770* [43], based on the *Haswell* microarchitecture. Figure 2.32 shows a sketch of the architecture of a single physical core. The *Haswell* microarchitecture, the successor of *Ivy-Bridge*, features 2 to 8 physical cores with hyper-threading support. Clock frequencies for this architecture are in the range of 2 GHz to 4 GHz. The memory interfaces are either dual-channel and quad channel, with theoretical peak bandwidth of 25.6 GiB/s (*i7-4770*) up to 68 GiB/s (*i7-5960X* [44], introduced June 2013). The *Haswell* microarchitecture doubled the SIMD floating-point performance with respect to its predecessor to 32 single precision-floating-point operations per second (FLOPSs) or two single precision AVX-FMA instructions respectively per cycle per core. When double precision floating-point values are used, half of this performance is reached. It supports SSE version 4.2 and and AVX version 2.

Hyper-threading

INTEL hyper-threading (HT) is a technology that allows hardware-based multi-threading. Each physical core is addressed by the operating system (OS) as two virtual cores. Instead of duplicating all resources, only some parts of the hardware are exclusively used by one virtual core (*e.g.*, register file). The remaining resources (*e.g.*, execution units, caches, system-bus interfaces, *etc.*) are shared among both. As the executing resources can be used in parallel, these can be used by another scheduled task. This does not increase the theoretical peak performance, but increases its utilization. Depending on the software, HT may have a significant performance increase, but additional OS scheduling and potential resource conflicts may also have a negative impact.

2.6.3. AMD *Piledriver*

The second CPU that was used in the course of this thesis was the AMD *FX-8320*, which is based on the *Piledriver* microarchitecture. Two physical cores of the CPU, so called 'integer clusters', are grouped into a 'module' with a shared floating-point unit (FPU). Figure 2.33 shows a sketch of the execution units of a module. The FPU is capable of processing two 128 bit vectors in parallel for SSE, but only a single 256 bit vector for AVX operations. In total, the two memory channels of the CPU have a peak bandwidth of 29.9 GiB/s.

2.6.4. Vc library

To write platform independent code while using the specific SIMD instructions supported by the target CPU, a library for explicit vectorization was used, the *Vc* library [117]. It allows to use an abstract class for vectorization without introducing any overhead. During compile-time the target platform is detected and optimizations are enabled accordingly. The library further allows for switching between different SIMD instruction set extensions,*e.g.*, AVX, SSE, or scalar implementations.

Version 0.7.1 was used during this thesis work. Various patches were created to add support for both platforms and their respective features. Especially for the platform detection code and support for AMD specific FMA-4 instructions.

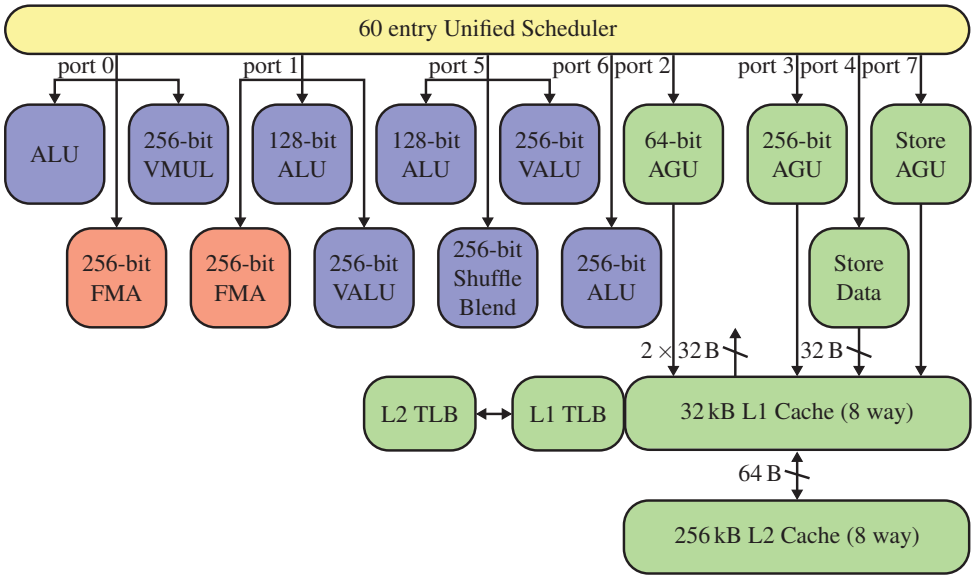


Figure 2.32.: Sketch of execution units and memory hierarchy of the INTEL *Haswell* microarchitecture. The FMA units are shown in red. Data related units are shown in green, others in blue. AGU denotes address generation unit. Adapted from `http://www.realworldtech.com/haswell-cpu/`.

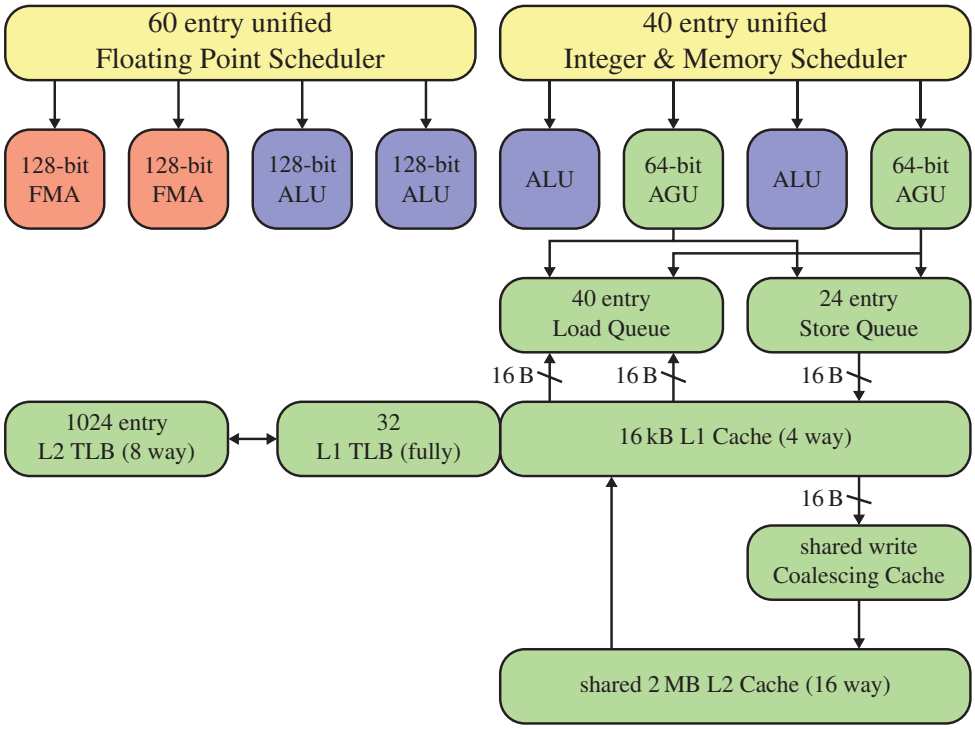


Figure 2.33.: Sketch of the execution units and memory hierarchy of a single module of the AMD *Bulldozer* microarchitecture. The sketch shows a single CPU core and the shared FPU (left hand side). FMA units are shown in red. Data related units are shown in green, others in blue. AGU denotes address generation unit. Adapted from `http://www.realworldtech.com/bulldozer/`.

2.7. Graphics processing units

The terminology in the following section is based on the NVIDIA *compute unified device architecture (CUDA) documentation [162].*

A GPU combines a graphics processor and dedicated graphics memory on a single PCB. The graphics processor contains several multiprocessors that allow each the execution of groups of threads in parallel (SIMT). A schematic of a typical GPU is shown in Figure 2.34a.

2.7.1. GPU computing

Originally, graphics processing units (GPUs) were designed to offload time-consuming graphics processing from the CPU to dedicated hardware. With the availability of programmable shaders with support for floating-point arithmetic, general purpose computation on graphics processing units (GPGPUs) became practically usable. The dedicated compute elements intended to transform geometry are well suited for matrix and vector operations. Manufacturers recognized the potential for GPGPU-based HPC and added additional features like shared memory and synchronization. Shaders were extended to so called 'compute kernels', which became programmable in *C*-like languages. Software development kits (SDKs) were released, that facilitated the programming of multi-threaded applications on GPUs.

For GPU programming, several frameworks exist. CUDA was the first, and is restricted to NVIDIA cards only. Other well known platform independent frameworks are for example the OpenCL and *DirectCompute* from MICROSOFT.

Specialized GPUs have been developed over the past few years to address the HPC market. They feature ECC memory and direct access of other PCIe devices (*GPUdirect*) and even lack a connector for a screen. Another drawback that was handled in the last years was the support for double precision floating-point operations. Being emulated in the beginning, current GPUs have dedicated hardware for double precision floating-point operations. But these are only accessible in the HPC version of the GPU and are disabled in the consumer grade 'low cost' equivalent.

Driven by the three-dimensional computer games mass market, the theoretical peak performance of GPUs nowadays is much higher than on any general purpose CPUs which leads to a very low price per FLOPS. Although GPUs run at much lower clock rates, the sheer parallelism of these devices massively accelerates parallel program execution.

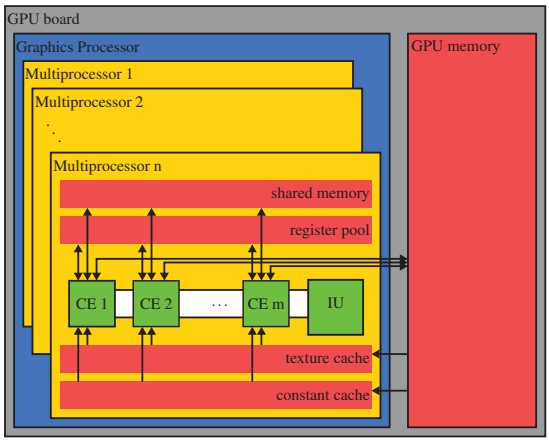
2.7.2. NVIDIA *GK110* architecture

The NVIDIA *GK110* graphics processor, that was introduced in February 2013, consists of 15 streaming multiprocessors (SMXs) and six memory controllers that achieve a combined bandwidth of the order of 300 GiB/s. A SMX, which is shown in Figure 2.35, comprises 192 single precision floating-point CUDA cores, 64 dedicated double precision floating-point units, 32 load / store units and 32 special function units used for transcendental functions. Threads are distributed across the units in groups of 32 threads, a so called 'warp'. Up to four warps may run concurrently, limited by the number of warp schedulers that are available per SMX on the chip. Each warp scheduler can dispatch two instructions per cycle. Therefore a maximum of 64 warps can be handled by the schedulers, allowing up to 2048 concurrent threads which all execute a single instruction. Such an architecture is called SIMT (*cf.*, section 2.5.1).

Since the launch of the *Fermi* architecture, even multiple compute kernels may run concurrently. Each SMX has a register pool comprising 65 536 registers, each 32 bit wide, that are shared among all warps. Additionally, up to 64 KiB of shared memory is available. The latter is shared with the 64 KiB L1 cache, allowing for different partitioning.

Due to the large register file, context switches can be done without overhead. Global memory access is done for half a warp in parallel, and can be done in one transaction (called coalescing) if the requested

(a) GPU schematic (architecture)



(b) GPU schematic (programming model)

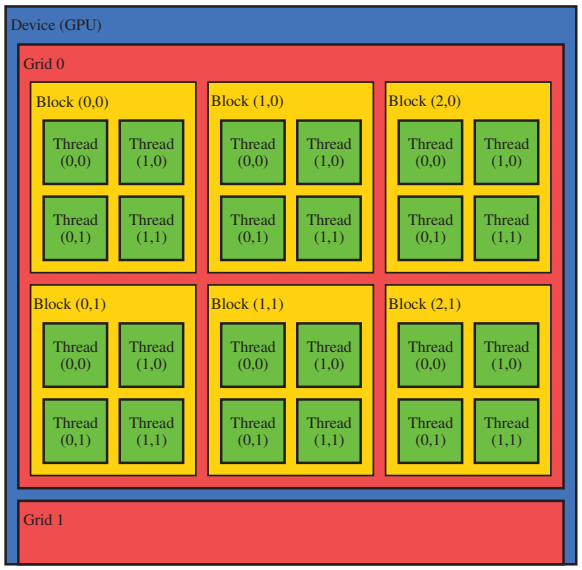


Figure 2.34.: Schematic of the architecture of current GPUs and the parallel programming model. (a) Graphics processor and attached GPU memory. Typically, for performance issues, memory is exclusively used by the GPU. For each graphics processor several multiprocessors are available. Each multiprocessor consist of a set of compute elements (CE) which are controlled by a common instruction unit (IU). Texture memory and constant memory reside in GPU memory but are accessed via individual caches. (b) A thread is executed as a grid of blocks of threads. The threads in a block are run synchronously. Blocks in a grid are scheduled individually. Multiple grids, or kernels can run concurrently on recent GPUs. Both, the grid and the blocks are depicted two-dimensional.

data is located in an aligned segment of the 16 data words (either 32 B, 64 B or 128 B, depending on the data type). If data is not aligned to segment boundaries, two transactions–one per segment–are required. On the other hand, branches within a compute kernel's code are expensive in view of cycles: if a thread diverges, each path is executed serially. Threads that do not need to be executed are transparently masked and excluded.

A summary of the features of a *GK110* based GPU, the EVGA *GTX 780Ti*, is shown in Table 2.8. In contrast to the professional grade TESLA cards, the card used in the course of this work is a consumer grade GPU with an artificially limited double precision floating-point performance (factor of 8).

base clock	876 MHz
streaming multiprocessors	15
CUDA cores / ALUs	2880 (192 per SMX)
peak performance (single precision)	2.5 Tflops/s (FMA instructions)
peak performance (double precision)	0.1 Tflops/s (FMA instructions)
registers per SMX	64 k (256 kB)
host interface	PCIe 3.0
graphics RAM	3 GB GDDR 5-SDRAM
memory bandwidth	336 GB/s

Table 2.8.: Properties of the EVGA *GTX 780Ti* GPU which is based on the NVIDIA *Kepler GK110* chip. Data taken from Refs. [1, 163].

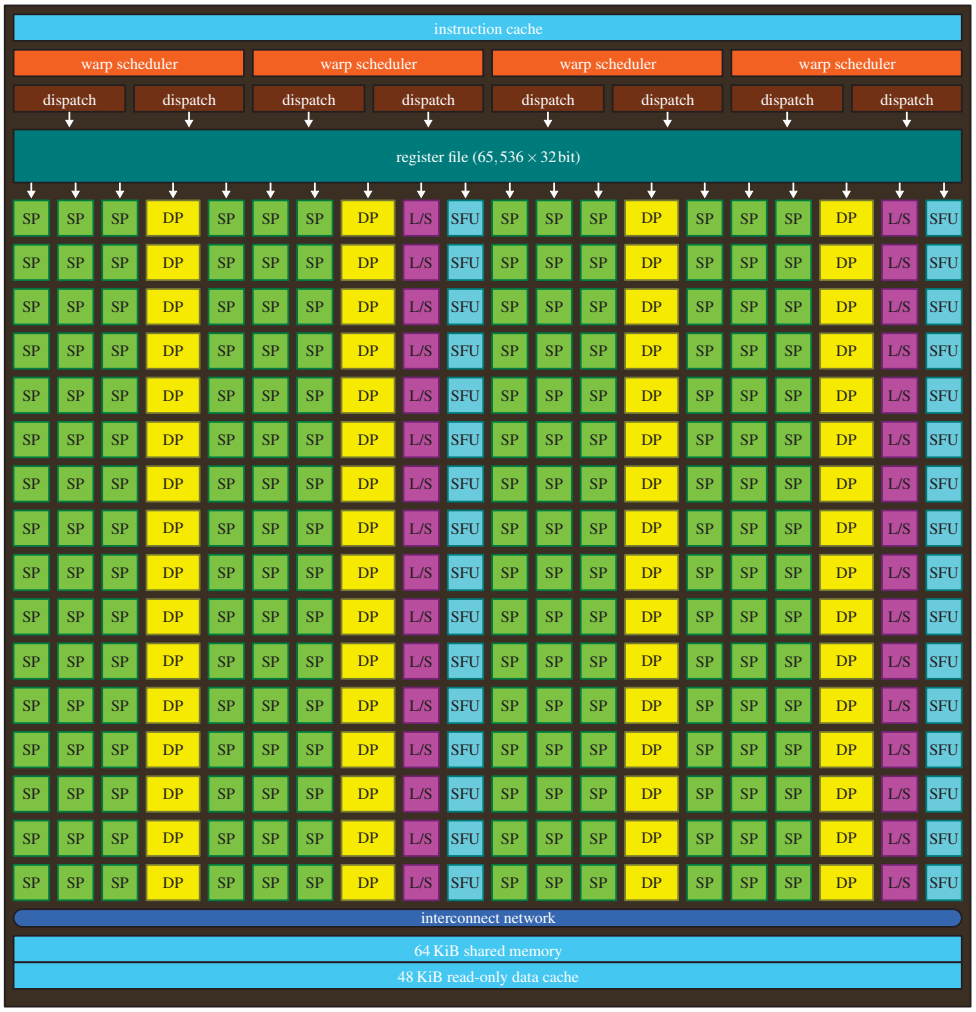


Figure 2.35.: Architecture of a single streaming multiprocessor (SMX) found on an NVIDIA *GK110* chip. Each SMX comprises 192 single precision (SP) CUDA cores, 64 double precision (DP) units, 32 special function units (SFU, *e.g.* for transcendental functions), 32 load/store units (L/S) and 4 warp schedulers with 2 instruction dispatchers each. A full *GK110* includes 15 streaming multiprocessor (SMX). Image adapted from Ref. [163].

```
// Kernel definition
__global__ void VecAdd(float* A, float* B, float* C){
  int i = threadIdx.x;
  C[i] = A[i] + B[i];
}

int main(){
  ...
  VecAdd<<<1, N>>>(A, B, C); // Kernel invocation with N threads
  ...
}
```

Listing 1: Illustration of declaration and invocation of CUDA kernels. In the shown example a vector addition is performed. `threadIdx.x` is the implicit thread index. Taken from Ref. [162].

2.7.3. NVIDIA CUDA

With the CUDA general-purpose parallel computing platform and programming model, GPU software is implemented as so called compute kernels or kernels for short. These are functions written in an extended *C* language. Kernels are invoked within the classic CPU code, and are then executed *N* times in parallel by *N* different CUDA threads on the GPU. An example is shown in Listing 1. In the programming model, as shown in Figure 2.34b, processing threads and data accesses are organized in 'blocks' and blocks are organized in 'grids', which may both be multidimensional (see Figure 2.34b for naming conventions). All threads have access to the same global memory. Shared memory is local to blocks and per thread up to 255 registers can be used. Each thread has access to an implicit index that is used to identify its position in a warp, block, or grid. This index is typically used to access the input/output data the thread operates on. Blocks may cooperate among themselves through barrier synchronization or shared memory. The size of a block must be set during compute kernel invocation. By setting the block and grid size, threads can be grouped easily to better match the underlying problem, even multidimensional blocks and grids are allowed.

Memory is transferred from the CPU to the GPU and back using the CUDA application programming interface (API). Kernels and data transfers may be parallelized using so called 'streams'. A stream is a sequence of commands of the CUDA API, *e.g.*, memory transfers or kernel invocation. Multiple of these streams may run concurrently. The API also supports explicit synchronization between such streams.

Kernel occupancy

Memory latency is one major cause for threads to stall execution. To hide the latency, the SIMT architecture supports fast switching between different warps with no overhead. To keep the hardware as busy as possible, it is advantageous to have a high number of warps ready to be scheduled on a single multiprocessor.

The ratio of currently active warps in a multiprocessor to the number of possibly active warps, the so called kernel occupancy, is used as a performance measure. Limiting factors for the maximum number of warps are, for example, the number of registers used by a thread. Here the number of active threads is crucial, which is 2048 at max for an SMX of the *Kepler* architecture (64 scheduler entries, 32 threads per warp, see above). The total amount of registers is limited to 65 536, thus if more than 31 registers are used per thread (plus one implicit for indexing) a full occupancy cannot be achieved. Other limiting factors are the amount of shared memory used per block as well as the size of the blocks. As up to 16 blocks per SMX are allowed, block sizes lower than 128 cannot reach full occupancy. Depending on the

kernel, whether it is compute- or memory-bound, the number of registers per thread can be traded for the number of active warps. Which may influence the kernel performance positively.

2.7.4. Benchmarking

As shown in Ref. [180], compute kernel execution times on NVIDIA GPUs follow an almost GAUSSIAN distribution. Already five runs give sufficient statistics to reach 0.2 % relative root mean square (RMS). Thus, all measurements shown in this work are mean values of five runs.

2.8. Comparison of the different platforms

Table 2.9 shows various performance measures of the platforms described above. If only the performance and memory bandwidth to the external RAM is considered, the GPU outperforms all of its competitors. A major advantage of the CPUs are the large caches that can compensate for the significantly lower memory bandwidth. The GPU on the other hand does not have large caches, and memory accesses have to be optimized to achieve a high performance. A key feature of the FPGA is its versatility. By designing long pipelines, memory accesses can be eluded.

Hardware	Chip	clock MHz	execution units	SP performance Gflops/s (FMA)	DP performance Gflops/s (FMA)	memory bandwidth GB/s
CPU	INTEL *Haswell i7-4770*	3400	64	217	109	25.6
CPU	AMD *Piledriver FX-8320*	3500	32	112	56	29.9
GPU	NVIDIA *GK110*	876	2880	2528	841[†]	336
FPGA	XILINX VIRTEX-6 *XC6VSX475T*	600	2016	1210[‡]	N/A	$\sim 105^{*}$

Table 2.9.: Theoretical peak performance of various architectures used in this thesis work. The number of execution units is given for SP units only. The given memory bandwidth considers the interface to external memory, only. Caches are not taken into account. In the case of the FPGA, the memory bandwidth depends on the actual design and external hardware. Also only the DSP slices contribute to the given performance. For both CPUs, each entry in a SIMD vector is counted as a core. [†]: This performance is not available on consumer grade hardware-based on this chip. [‡]: Natively, no floating-point support is available, thus the performance measure accounts for integer operations only. [*]: Assuming that all 840 IO-pins are used to interface memory. In the highest speed-grade, up to 1 Gbit/s per pin can be achieved [249].

3. State of the art

In this chapter, starting with an overview of fast scientific cameras, the state-of-the-art in correlator design is discussed. An introduction of the SPIM-FCS technique was already given in section 1.1.2 and correlation algorithms were already discussed in section 2.4.1. The following section contains a comparison of several detectors that are used for SPIM-FCS. An introduction to fast signal correlators that are suitable for FCS is given in the subsequent section.

3.1. Detectors for imaging FCS

Fast cameras with a good sensitivity are required to detect the movement of small particles in solution entering and leaving the focal volume of a microscope. These detectors also have to provide a sufficient number of pixels to allow imaging of larger structures, *e.g.*, cells. Therefore single-line detectors are impractical. Figure 3.1 shows the performance of several scientific cameras that were commercially available in 2013. Additionally, the *RADHARD2* single photon avalanche diode (SPAD) array is shown. The figure shows results of 100 nm fluorescent microspheres as a medium sized sample. Table 3.1 summarizes the features of the different detectors. The plots show that only the *RADHARD2*, the *SA-05* and the *ORCA-Flash4.0* were able to resolve the plateau of the correlation curve that is necessary for a reliable determination of the particle concentration. From those three detectors, the *RADHARD2* showed the lowest noise.

Next to speed, the sensitivity of the detector is crucial, because single fluorescent molecules have to be detected. In this discipline, electron multiplying charge coupled device (EMCCD) cameras proved advantageous. In case of the back-illuminated *iXon X3 860*, a quantum efficiency (QE) of above 90 % is reached with a very low readout noise [11]. Front illuminated cameras are typically 30 % less sensitive (see section 2.1.2 for a comparison). The active area of SPAD arrays has a comparable performance in the blue to green spectrum. However, only a small part of a single pixel can be used for light detection, thus limiting the sensitivity even further.

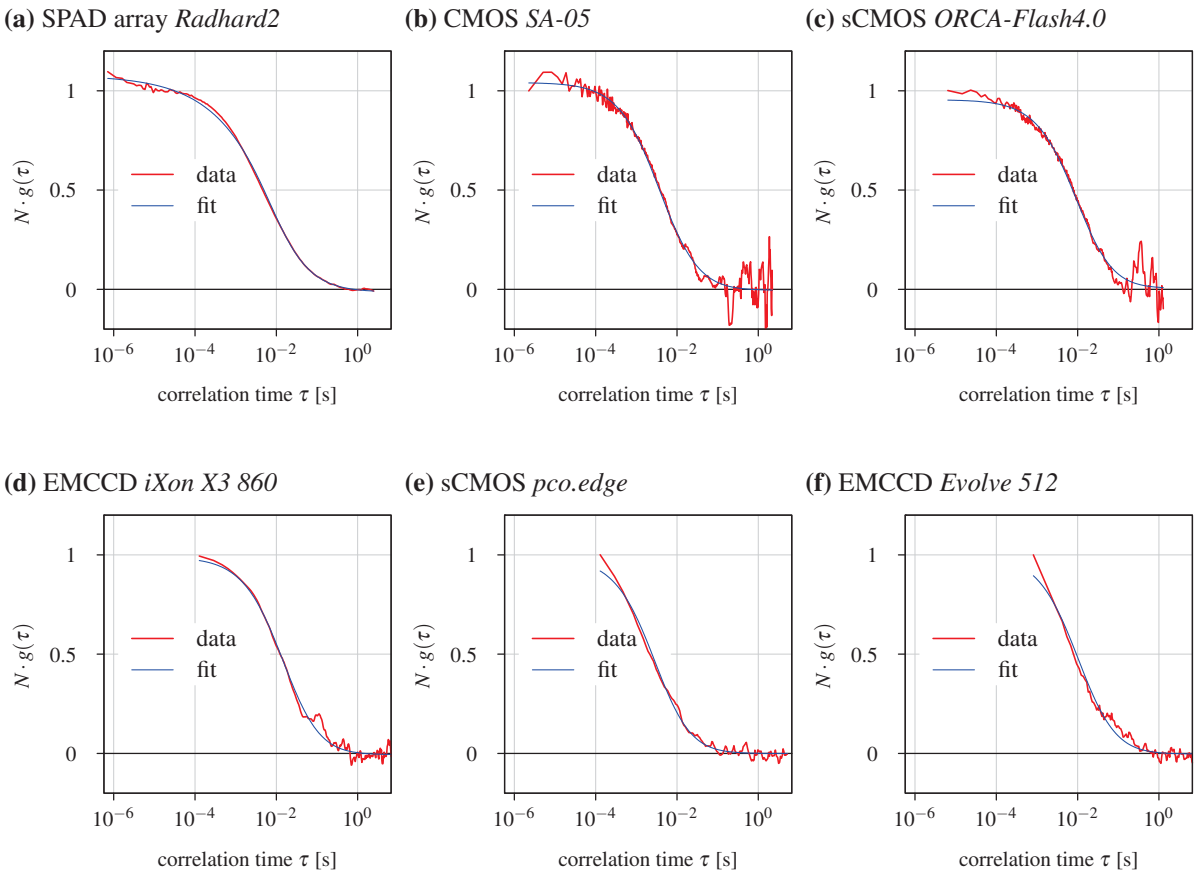


Figure 3.1.: SPIM-FCS autocorrelation curves of 100 nm **green fluorescent microspheres, obtained with several different cameras.** The detectors were read out as fast as possible for the given ROI. See Table 3.1 for the parameters of the sensors. These results were already published in Ref. [201].

	Radhard2	*SA-05*	*ORCA-Flash4.0*	*iXon X3 860*	*pco.edge*	*Evolve 512*
technology	SPAD array	CMOS	sCMOS	EMCCD	sCMOS	EMCCD
QE @ 525 nm [%]	35	35	70	95	54	95
pixel size [μm]	4	20	6.5	24	6.5	16
pixel pitch [μm]	30	20	6.5	24	6.5	16
exposure time [μs]		16.6	38.80	450	486	2000
cycle time [μs]	3	16.8	38.95	489	495	2380
ROI [pixels]	8×32	20×20	6×100	32×32	20×20	20×20
ROI [μm^2]	240×960	400×400	39×6500	768×768	130×130	320×320

Table 3.1.: Comparison of various parameters of different cameras evaluated for imaging fluorescence correlation spectroscopy. Exposure time and cycle time are given for the specific ROI. For simplicity, QE and PDP are equated. Adapted from Ref. [201].

3.2. Fast signal correlators

The basic task that a correlator performs is to multiply two discrete intensity values and to accumulate the result for each lag time τ; a multiply accumulate (MAC) operation (see equation (1.10)). In case of a multiple-τ correlator with the binning ratio $m = 2$ and $L = 8$ lags per block, 16 MAC operations have to be executed for every input value (see section 2.4.2). Therefore the speed of a correlator mostly depends on the performance and the parallelizability of the MAC-operation. Correlators that are based on field programmable gate arrays (FPGAs) typically implement these operations in discrete digital signal processor (DSP) cells which can run in parallel. Implementations for either CPUs or graphics processing units (GPUs) are commonly based on the floating-point data type. Hence, a MAC operation is performed as two single floating-point operations (FLOPs), or if supported, a single fused multiply accumulate (FMA) FLOP.

Throughout this work the performance is given in FMA-flops/s. In case of correlator implementations that are not based on intensity time series, but rather on photon arrival times or inter-photon times, an equivalent performance of a comparable multiple-τ correlator is given.

Michalet et al. [151] already proposed that dedicated digital signal processors, FPGAs or GPUs are needed to achieve real-time processing of data, obtained with imaging sensors. In the following, several hard- and software implementations of correlators are discussed. Most of them have been developed especially for the application in confocal fluorescence correlation spectroscopy (FCS) and dynamic light scattering, therefore they were no option for the large amount of pixels that are required in imaging FCS.

Hardware correlators

First correlators were built as application-specific integrated circuits (ASICs) [188]. Since the availability of FPGAs, most hardware correlators are based on reconfigurable hardware. To further improve the performance, ASICs have still been used as front ends [62, 90]. Several correlators, and especially the commercial hardware correlators, are used as raw data acquisition devices, too. The latter are either available as PC add-on cards or USB devices, supporting up to 32 channels [5, 46].

Hardware correlators are mostly based on the multiple-τ algorithm. Either using a parallel implementation of linear blocks (see section 2.4.2) [73, 90, 102], or a serial multiple-τ scheme (see section 2.4.2) [72, 105, 135, 152, 155]. Resource sharing, *e.g.*, of DSP blocks within the FPGA, is mostly used for the less often executed higher order blocks [101, 103, 105, 152, 155]. More advanced implementations use up to three different designs for the linear blocks [72]. First blocks are typically made of shift registers, this is particularly efficient because single-bit-multiplication becomes a simple `and`-operation. Blocks in the middle of the correlator use dedicated DSP-slices, whereas blocks at the end of the correlator share resources with neighboring pixels.

A completely different approach is followed in Ref. [93]. Here a logarithmically spaced correlator with a single-bit data path is proposed. To span several decades of delay values, signals are delayed using on-FPGA FIFOs.

Table 3.2 compares a variety of implementations that were published before and during this work. The FPGA-based implementation described in section 5.5 is referred to as Ref. [155] Prior to the start of this work no hardware correlator was available that could deal with more than 32 channels in parallel. At the same time, the commercial solutions were on average an order of magnitude faster than the published implementations. From the perspective of total performance, no implementation was able delivering 164 Gflops/s, that is required for real-time correlation of the CHSPAD's raw data.

Software correlators

CPU correlators. To not depend on dedicated correlator hardware, several software implementations have been proposed that can run on standard PC hardware. Table 3.3 gives details on the different

publication	type	platform	res.	ch.	lag org.	hardware	perf. [Gflops/s]
Schätzel [188] (1985)	multi-τ	ASIC	5.3 µs	1	8 or 16	"*ALV-3000*"	0.006
Engels et al. [62] (1999)	multi-τ	ASIC	5.0 ns	1	32	"*CORR32*"	6.6
Hoppe et al. [91] (2001)	multi-τ	ASIC	6.26 ns	1	16/8	N/A	3.8
Jakob et al. [102] (2007)	multi-τ	FPGA	10 ns	1	16/8	N/A	2.4
Liu et al. [135] (2008)	multi-τ	FPGA	1 µs	1	8	*XC2VP100*	0.016
Mocsár et al. [152] (2012)	multi-τ	FPGA	400 ns	4	8	*XC2V3000*	0.160
Buchholz et al. [155] (2012)	multi-τ	FPGA	10 µs	1024	8	*XC2VP40*	1.6
Kalinin et al. [105] (2012)	multi-τ	FPGA	4 ns	2	8	*XC6SLX45T*	8
Hromalik et al. [93] (2013)	log-τ	FPGA	100 ns	256	40 tot.	*XC6VFX550T*	(41)
Gong et al. [72] (2014)	multi-τ	FPGA	100 ns	32	16/8	*XC6SLX150T*	7.7
Gong et al. [73] (2014)	multi-τ	FPGA	10 ns	1	16/8	*XC6SLX150T*	2.4
ALV7004/FAST [5] (2007)	multi-τ	FPGA or ASIC	3.125 ns	4	16/8		30.7
ALV7032 [5] (2011)	multi-τ	FPGA or ASIC	50 ns	32	16/8		15.4
Flex01LQ-05 [46] (<2010)	multi-τ	FPGA	5 ns	1	64/32	"*MT-64*"	19.2

Table 3.2.: Comparison of different hardware correlators. For comparison, commercial products are shown in the lower part of the table. FPGA implementations are mostly based on XILINX chips, starting with *XC*. If different correlator architectures are combined, *e.g.* FPGA plus CPU, only the front end correlator is considered. *ch.* represents the number of independent channels. *lag org.* represents the organization of lags. If two values are given, the first is the number of lags in the first linear block and the second value the number of lags for all other blocks. Performance measures given (*perf.*) are either based on the actual implementation or, if denoted in braces, assuming an eight lag multiple-τ correlator with the same temporal resolution. The value is given in Gflops/s (FMA). The thick line indicates the start of this work.

publication	type	platform	res.	ch.	lags	software	perf. [Gflops/s]
Magatti and Ferri [140] (2001)	multi-τ	INTEL P3 550MHz	5 µs	1	28	*LabView*	0.01
Magatti and Ferri [141] (2003)	m-τ + PM	INTEL P4 1.5GHz	30 µs	1	28	*LabView*	0.02
Wahl et al. [226] (2003)	PM	AMD 1800+ 1.5 GHz	10 µs	1	—	*C*	(0.002)
Laurence et al. [128] (2006)	PM	INTEL P-M 2 GHz	4.6 µs	1	—		(0.003)
Lee et al. [129] (2009)	PM	1/4th of quad-core	<0.1 µs	3	—	*C*	(0.48)
Tieman et al. [217] (2011)	multi-τ	compute cluster, 2 cores	<1.7 s	10^6	?	*MPI*	(0.01)
Schaub [191] (2012)	PM	INTEL i7 Q720 1.6 GHz	0.1 µs	1	—		(0.16)
Vitali et al. [222] (2014)	multi-τ	NVIDIA *GeForce GTX 780*	31 µs	1024	8	CUDA	0.5

Table 3.3.: Comparison of different CPU and GPU correlators. For photon-mode (PM) correlators, the resolution given is the time needed to process a single photon. *ch.* represents the number of independent channels, or – if used for imaging fluorescence correlation spectroscopy (imaging FCS)– the number of pixels. *lags* represents the number of lags in a linear block. Performance measures (*perf.*) given are either based on the actual implementation or, if denoted in braces, assuming an eight lag multiple-τ correlator with the same temporal resolution. The value is given in Gflops/s (FMA). The thick line indicates the start of this work.

implementations and their performance. A major advantage of CPU-based correlators is their ability to improve performance by buying new hardware or using multiple computers in parallel. Furthermore, the computers can also be used for visualization and data storage.

Fast CPU correlators typically fall into two main categories: the first group is based on photon arrival times, working in so called photon-mode (PM), the second relies on intensity time series that can be calculated by accumulating light, *i.e.*, photons, in fixed time-bins. PM correlators typical need a constant time per photon event and thus work well in low-light situations, *i.e.*, they are $\mathcal{O}\left(N_{\text{photon}}\right)$ [128, 129, 191, 226]. If the photon rate exceeds a certain threshold, they loose the ability to process data in real time. Their temporal resolution is typically limited by the accuracy of the time-tags of the photons.

Most of the correlators, that fall into the second category, need a constant time to process a single frame or time-bin [140, 141, 217]. Such correlators are typically based on the multiple-τ algorithm and introduce a certain averaging for larger lag times, which is not necessarily a disadvantage for FCS. The multiple-τ algorithm, with an algorithmic complexity of $\mathcal{O}\left(N_{\text{frames}}\right)$, allows for easy parallelization either on the block-level [49] or per pixel on a computer cluster based on *MPI* [217]. A special position is taken by the correlator described in Ref. [141] which uses a combination of both: For small lag times a multiple-τ correlator is used, whereas longer lag times are calculated in PM. A different approach based on fast Fourier transform (FFT; WIENER-CHINTCHIN-theorem [109]) is described in Ref. [125].

The overall performance of the software correlators is significantly below that of the commercial hardware correlators, but can constantly be increased with newer hardware. Only one implementation exists, that can deal with many pixels as required for the Swiss single photon avalanche diode array (CHSPAD) array. Unfortunately, the performance was three order of magnitude below the requirements. The table further shows that when more than three channels are processed, usually the multiple-τ correlation algorithm is used.

GPU correlators. Graphics processing units (GPUs), that are particularly optimized for MAC operations, seem to be a promising platform for correlator development, too. Their massive parallel structure is ideal for parallel correlation of thousands of pixels. So far, only two implementations have been proposed [127, 222] (performance details were only available for the second implementation). Table 3.3 gives an overview of the performance details. Although the described correlator offers 1024 channels, which would match the *RADHARD2* SPAD array, the overall performance is lower than recent FPGA-based implementations. And a factor of three below the requirements of the *RADHARD2*.

Conclusion

At the beginning of this thesis (mid-2010), no solution to calculate correlation estimates for several hundred pixels in parallel was available. Even from the perspective of performance none of the implementations fulfilled the requirements of the CHSPAD. Initially, the most promising platform seemed to be an FPGA as hardware-based implementations typically outperformed every software correlator. Therefore the FPGA was chosen as a first platform for a new design of a massive parallel correlator. Later, recent developments in the field of CPU and GPU computing made these platforms particularly attractive for correlator design, too. Especially on those platforms much shorter development times are expected.

4. SPAD arrays for imaging FCS: readout design and performance evaluation

This chapter introduces the two SPAD arrays that were evaluated in this work as well as the hardware used for their readout. First the *RADHARD2* sensor is described in section 4.1 and the newer CHSPAD sensor in 4.2. The chapter closes with a characterization of the CHSPAD array in section 4.3.

4.1. The *RADHARD2* single photon avalanche diode array

The first sensor that was assessed and integrated into the SPIM-setup was the *RADHARD2*, that was available from the beginning of this work. This sensor was designed in EDOARDO CHARBON'S lab (TU Delft, Netherlands and EPFL, Switzerland) and was first published in 2009. An in-depth documentation of the *RADHARD2* can be found in Refs. [35, 36].

4.1.1. Sensor hardware platform

The *RADHARD2* sensor is supplied soldered onto a PCB (printed circuit board, the so called 'daughter board') that fits onto the connectors of the '*LASP*' FPGA development board (EPFL, Switzerland; see appendix B.1 for further details). This development board carries two VIRTEX 2 field programmable gate arrays (FPGAs) along with the necessary peripherals and a small amount of static random access memory (SRAM). Both FPGAs are linked by a 100 bit wide parallel interconnect. Data transfer to a host computer was realized using the on board CYPRESS *EZ-USB* USB 2.0 controllers with the provided firmware. The board features two USB ports, one connected to each of the two FPGAs. The 48 MHz clock provided by the USB controllers, was used as a clock source for the entire hardware design.

4.1.2. Readout controller

The sensor was shipped with a rudimentary firmware that allowed capturing single frames. For SPIM-FCS, a complete redesign was done to also support fast raw data acquisition. The 'readout controller' core generates the signaling, required by the *RADHARD2* single photon avalanche diode (SPAD) array, and reads out the sensor in rolling shutter mode (*cf.*, section 2.1.1). The acquired raw data is then transferred to a host computer via USB. Table 4.1 summarizes the properties of this core. During readout, a row is selected, read and reset. This scheme is shown in Figure 4.1. This takes four clock cycles of the main clock and in sum 2.7 µs for the entire SPAD array at 48 MHz. Less time is required when only a subregion is read. The exposure time can be extended from 2.7 µs by introducing additional wait-cycles

property	value
base clock of readout hardware	48 MHz
minimum full-frame integration time	2.7 µs
maximum frame rate	370 kfps
maximum data rate (incl. frame overhead)	49.6 MB/s
minimum integration time for subregion	169 ns (2×32 pixel)

Table 4.1.: Properties of the *RADHARD2* detection system.

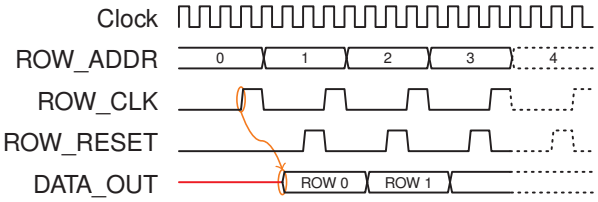


Figure 4.1.: Timing diagram of the readout procedure of the *RADHARD2* SPAD array. Signals ROW_ADDR, ROW_CLK and ROW_RESET are driven by the FPGA. DATA_OUT is connected to input pins of the FPGA. The clock of the controller core is shown for reference. Constantly driven signals are omitted. Signal names are according to Table 2.2.

between two readouts. A further reduction of the readout scheme towards three or fewer cycles or higher clock rates might allow for shorter frame integration times, but this was not tested.

Various aspects of the controller can be configured during runtime via USB by writing to several registers of the controller core. These registers allow setting a region of interest (ROI), to start and stop the acquisition, to set the exposure time of a single frame, and to enable or disable temporal binning for the correlator (see below). This is fully supported in the *QuickFit3 RADHARD2* camera plug-in.

Additionally, the readout controller adds start-of-frame and end-of-frame markers. Data is then fed into two data paths: one path towards the correlator within the second FPGA and another one to the USB-interface. The latter encapsulates raw data into frames by adding a 32 bit header containing the frame number and a start-of-frame marker (`0xFF`) as well as a 16 bit cyclic redundancy check (CRC) checksum (standard *CRC-CCITT* polynomial `0x1021`). After being packed into a frame format (see below), data is pushed to the USB interface controller. At this stage no further buffering is done, so if the USB controller stalls, data is lost.

The final data rate at 10 µs integration time (134 B per frame) is 12.8 MB/s. Although this should easily be handled by a USB 2.0 connection (max. bandwidth $\sim$ 40 MB/s), data loss occurred above that rate. Therefore, frame-based error checking was introduced, which allows detecting missing or broken frames based on CRC. Probably, retransmits during bulk transfer caused the USB controller to stall in this situation.

Temporal binning

At high fluorescence intensities, the probability of more than one photon event during the integration time of a single pixel increases (this effect can be seen in Figure 4.16 for the CHSPAD if the count rates exceeds 1 kHz). To effectively reduce the probability to miss photons, so called ‘temporal binning’ can be enabled on demand. If enabled, three consecutive frames are added up pixel-wise in the read-out controller core. Thus, a three fold higher frame rate can be used although the hardware correlator (see section 5.5.2 for details) is limited to 100 kfps.

4.1.3. System overview

Figure 4.2 gives a schematic overview of the final design for the *RADHARD2* SPAD array. The left part of the image shows the *LASP* development board with the daughter board and the *RADHARD2* on top. The hardware design in the FPGAs is depicted in the grayish area. External memory used for data reorganization and context store is shown in light yellow. Data acquisition is mainly done in the first FPGA. The design of the second FPGA comprising the correlator is described in more detail in section 5.5. Each of the two FPGAs use a dedicated USB interface to the host computer, which allows data transfer of the raw data as well as remote control of the firmware.

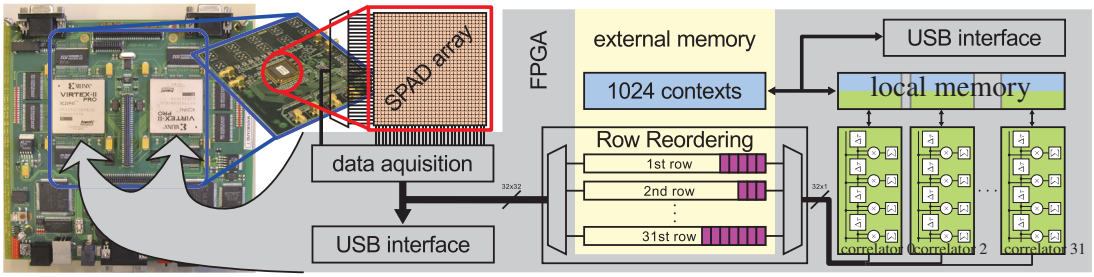


Figure 4.2.: System layout of the *RADHARD2* hardware design. On the left, the *LASP* main board is shown which holds the daughter board (blue inset) with the *RADHARD2* SPAD array (red inset). The FPGA design is shown in gray. Data acquisition shown on the left hand side of the design is done within the first FPGA, the right hand side comprises the hardware correlator (*cf.* section 5.5.2), that is implemented in the second FPGA. The correlators are shown in green. External memory (yellow) is used for storing each pixel's context (light blue).

4.2. The Swiss single photon avalanche diode array

In 2012, a more advanced SPAD array, the Swiss single photon avalanche diode array, became available. As the *RADHARD2*, this sensor was also incorporated into the microscope. Due to its significantly larger size and different pixel structure, the readout had to be redeveloped. This sensor was also designed in EDOARDO CHARBON'S lab and was first published in 2013. Details on the Swiss single photon avalanche diode array (CHSPAD) can be found in Refs. [29, 31, 33].

This section focuses on the design of the readout controller. A more detailed characterization of the entire system comprising sensor, the FPGA firmware, and the optics is given in section 4.3.

4.2.1. Hardware platform

Figure 4.3 shows an overview of the hardware platform used for the CHSPAD. The *Broaddown 4* FPGA development board (Enterpoint Ltd., Malvern, United Kingdom) forms the basis of the system (see appendix B.2 for details). It features two XILINX VIRTEX 4 FPGAs as well as DDR 2 memory and various peripherals. The daughter board holding up to two CHSPADs is directly attached to the development board. Each of the two possible sensor positions is controlled by a single FPGA. Both VIRTEX 4 are connected by set of differential pairs. Unfortunately, the board itself lacks any high-speed interface to connect to the outside world. Thus, a connection to the host computer was realized again with a CYPRESS *EZ-USB* USB 2.0 controller attached to one of the pin headers.

Originally, a custom firmware was designed for real-time data acquisition. It was implemented based on the limited information that was available on the CHSPAD at that time. This implementation, mostly

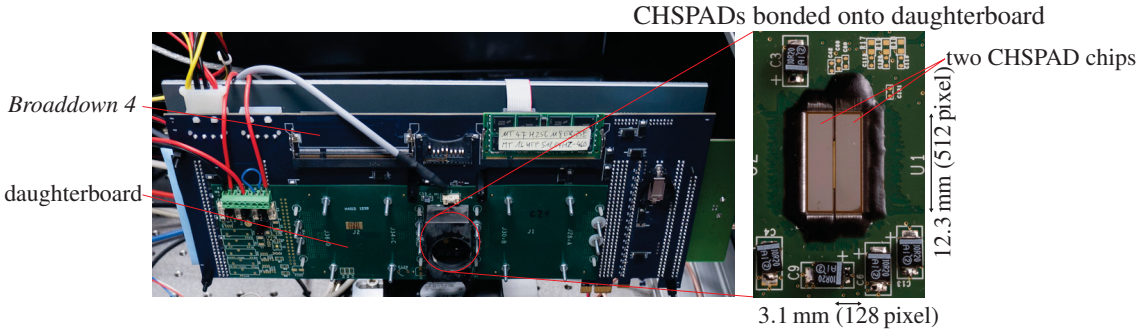


Figure 4.3.: Overview of the CHSPAD hardware platform. The left part shows the *Broaddown 4* FPGA development board (blue) including the daughter board (center, green). The CHSPAD chip(s), as shown on the right, are bonded directly onto the daughter board.

based on the *RADHARD2* readout design described above, proved to have some disadvantages regarding the active recharge and was therefore not used for sample measurements. As it offers real-time data acquisition, it is described in section 4.2.5. Sample measurements as described in chapter 6, were carried out using a modified version of the provided firmware, which is described in the following.

4.2.2. Provided readout firmware

The readout firmware provided for the CHSPAD by SAMUEL BURRI (EPFL, Switzerland) was only able to readout full frames. Due to a lack of high-speed interfaces, raw data was written to 2 GiB on board DDR-2 memory. After the measurement, data was transferred to the host computer via a USB 2.0 interface.

In each clock cycle, one forth of a line can be read (4-to-1 multiplexer, see section 2.1.4 and Figure 2.12 for details), so 512 clock cycles are required for reading a full frame. To increase the frame rate, readout is done in a rolling shutter mode. The readout process itself is fully independent of the electrical driving of the diodes (*e.g.*, active recharge) to reduce artifacts that might occur due to correlation of both signals: If for example a recharge pulse, which is applied globally to all SPADs, occurred once per frame, a gradient would be visible, as the active period of the SPADs differs between lines. Thus, to reduce these artifacts, the controlling part and the readout part of the CHSPAD are located in different (mostly unrelated) clock domains.

4.2.3. Enhancements of the readout firmware

To fulfill the requirements of a detection system for SPIM-FCS, several features were added to the original readout firmware: The memory on the development board was increased to 4 GB DDR 2 RAM to allow measuring for a longer duration. Also the clock speed of the firmware was increased to 80 MHz, which is the maximum possible clock speed for the CHSPAD and increases the frame rate to 156 250 fps. Further modifications are described in the following.

Broken pixel removal

The CHSPAD sensor with microlenses (chip *C21*) had $\sim$ 1800 broken pixels, which were $\sim$ 3 % of all pixels (a map of the broken pixels is shown in Figure 4.5). The majority of these defective pixels tended to detect photons in most of the frames and might have suffered from short circuits or other defects in the silicon substrate. Still others were over-proportionally dark and have probably been covered by dust particles. Another possibility for defective pixels were defects in the microlens coating.

Those broken pixels were sorted out by hand in the control software, based on the measured gray values at different light intensities. The aim was to have a homogeneous field of view for a homogeneous illumination.

To achieve homogeneous images, broken pixels were removed from the images by filling in the missing information from the surrounding pixels. This can either be done in software after data has been transferred to the recording computer (*e.g.*, by using an average over the surrounding 'good' pixels in a 3×3 neighborhood), or directly in hardware. The latter option is extremely efficient if large image stacks are recorded since no further data processing step for defect removal is required after recording. Also when data compression based on entropy encoding is used (see below), pixels that do not follow the majority statistics may have a strong negative impact on the overall compression efficiency.

In hardware, the averaging process is done in a five stage process by randomly selecting either a left or right 'good' pixel for every broken pixel from an external map of defective pixels. Figure 4.4 shows the corresponding network. The mask of defective pixels is loaded by the control software into the FPGA firmware and propagates along with the actual pixel information through the averaging stages. In the first three stages randomly replace a masked pixel by a 'good' pixel in a horizontal neighborhood of

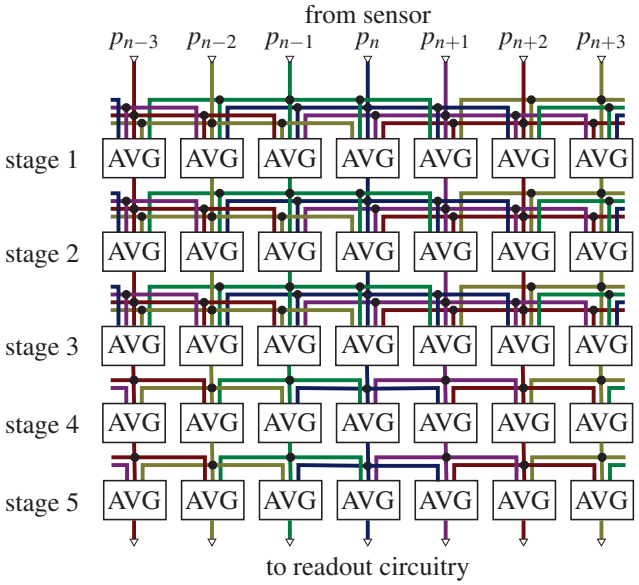


Figure 4.4.: Schematic drawing of the five stage averaging process for broken pixels. The first three stages, averaging units ('AVG') randomly select a 'good' pixel in a five pixel neighborhood, if the center pixel p_n is masked broken. In total, horizontal areas of up to eleven masked pixels can be averaged with 'good' pixels. The two last stages randomly choose the left or the right pixel if the center pixel is masked. Data flow is from top to bottom. Beside the value of a pixel, also the mask information is passed through the stages.

five pixels. The last two stages randomly replace a masked pixel by its left or right neighbor. With this scheme, up to 11 masked neighboring pixels can be replaced.

In each stage a linear feedback shift register (LFSR) with a unique initial value calculates the required random numbers. The random replacement does not impose an any further correlated signal that might distort the correlation curves in an FCS analysis.

Region of interest

The provided design was only able to record full frames. Considering the limitation of 4 GiB of DDR 2 SDRAM of the development board, measurements were limited to less than four seconds. This is typically too short for proper fluorescence correlation spectroscopy (FCS) measurements. But as single biological cells only cover a small part of the sensor, a valid method to reduce the overall amount of data and at the same time to extend the duration of the measurement is to discard information of unneeded pixels.

To keep the complexity of the hardware implementation as low as possible, a ROI cannot have arbitrary size. Vertically, it can start at any line, but horizontally only blocks of 128 pixels are allowed, starting at x-axes position $x = n \cdot 128$ with $n = 0,1,2,3$. Typical animal cells (*e.g.*, HeLa-cells, see section 6.9.3) are smaller than the 128 pixels ($\approx 51.2\,\mu$m at $60\times$ magnification). Due to these limitations, it is important to position the sample within one of the four blocks of 128 pixels, which is easy to do on the selective plane illumination microscopy (SPIM) setup. The raw data stream from the sensor consists of multiplexed 128 bit data words. In a first step, these are de-multiplexed and repacked to 128 bit wide words containing pixels in the correct order. Data words of 128 bit in a single frame are then accepted and discarded based on their number ($0 \ldots 511$) for a rough selection of the ROI. Additionally, unused columns in between are cut out. The settings of the ROI are written to a configuration file that is interpreted by the CPU- and GPU-based correlators, so that they can rebuild a proper image from the raw data stream.

Readout software

To control the firmware and to display a live-view from the sensor, a graphical user interface (GUI) software, *ngsoft*, was used. The software was provided along with the firmware by SAMUEL BURRI (EPFL, Switzerland). Several modifications were applied to this software to account for the changes in the firmware. Support for remote control was added via a TCP/IP server component. It gives *QuickFit3* full control over the software and the sensor (*QuickFit3*-client-plugin, development was done in cooperation with JAN W. KRIEGER, DKFZ Heidelberg, Germany). Further view modes were added to *ngsoft*, to support the experimenter during the alignment of the instrument and for automatic contrast enhancement. Figure 4.5 shows and image of the GUI.

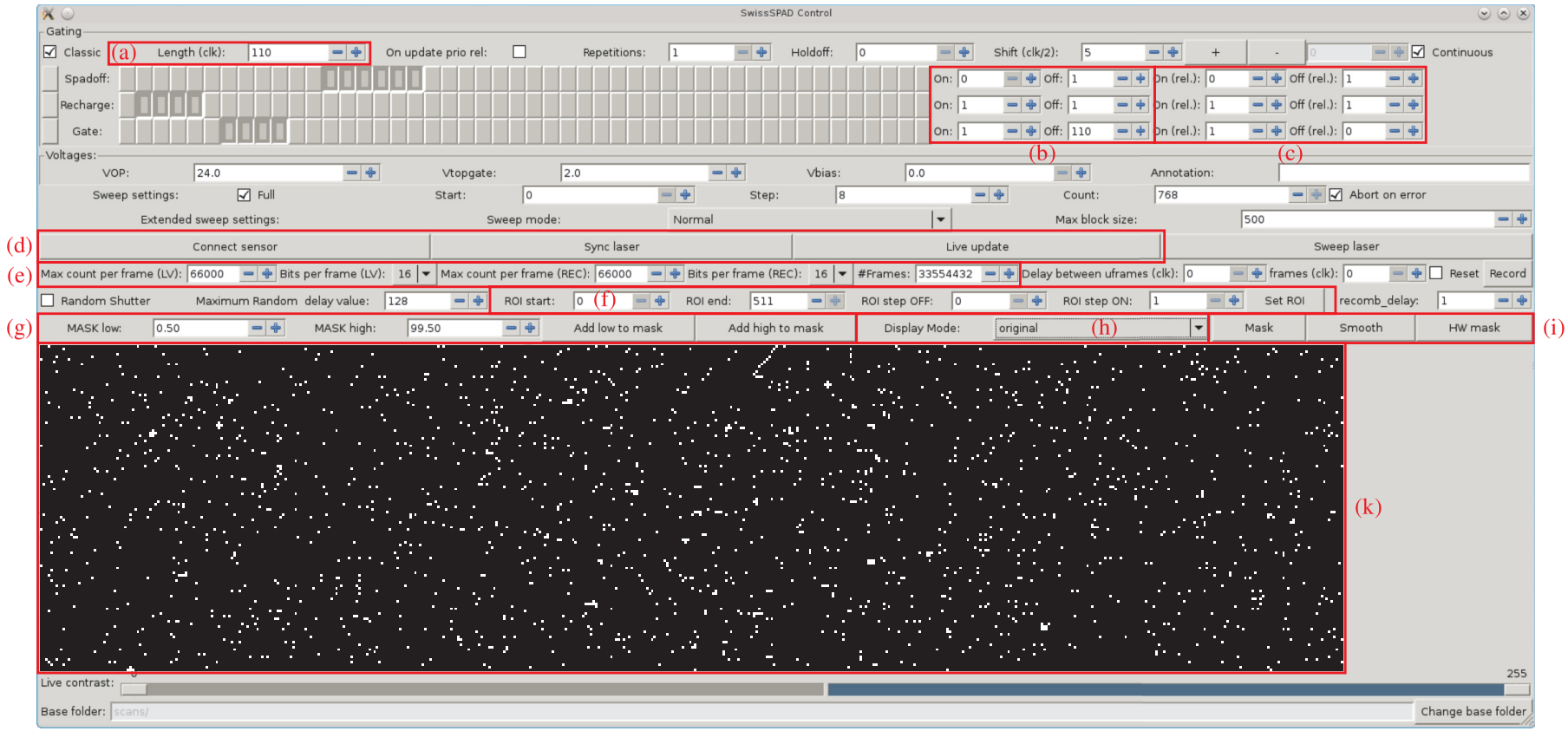


Figure 4.5.: Screenshot of CHSPAD control software. (a) Duration of the gating scheme in clock cycles. (b) Switching positions for SPADOFF, RECHARGE and GATE in absolute values. (c) same as before in relative value to allow simple changing of the entire duration. (d) Sensor controls. *Connect sensor* establishes an USB connection to the sensor hardware. *Live update* enables live view. (e) Select bit-width of real time image and for recording. (f) ROI control. (g) Software assisted broken pixel selection. (h) Various display modes (auto-contrast) to support alignment. (i) Mask control. *Mask* shows the mask in the real time image as an overlay. *Smooth* removes masked pixel by smooth operation in software. *HW Mask* enables smoothing in hardware. (k) Real-time image taken with sensor. In here, the defective pixels are shown.

4.2.4. Reduction of afterpulsing

Initially, the CHSPAD was controlled with a signal pattern as proposed by the chip designer (SAMUEL BURRI, EPFL, Switzerland). Unfortunately, this control scheme led to a significant afterpulsing in low-light conditions (see also section 2.4.3). Figure 4.6a shows an average autocorrelation function for a measurement on a darkened sensor (sensor *D1*, see section 4.3 for details on the sensor). For an ideal sensor, a flat correlation curve is expected (white noise has an autocorrelation function (ACF) $G(\tau) \propto \delta(\tau)$). Contrary to this expectation, the autocorrelation curve showed a significant low-lag amplitude that decays towards large lag-times τ. The afterpulsing can be divided into two regimes, one fast decaying "strong" component, which was only visible in the first channel, and a slow decaying "weak" component that was visible on the microsecond to millisecond time-scale (see Figure 4.6a). The strong component can be explained by trapped charges in the silicon substrate that cause a second avalanche in the diode [61]. These are typically removed by discarding the first channel of an autocorrelation curve (ACC). The weak component with a much longer decay time is still not fully understood. A possible cause could be the correlation of the CHSPAD signaling and the readout.

A reduction of the afterpulsing is essential for successful FCS measurements, since the decay-time of the afterpulsing is of the same order as the diffusion time of many molecular probes (*e.g. Alexa-488*, see section 2.2.4).

Control signal schemes

To reduce the afterpulsing at least to some extent, different signal patterns were tested to drive the CHSPAD. Figure 4.6b shows the results for the mean autocorrelation curve for different control schemes *1*, *2*, *3*, *4*, and *8*. The parameters of the schemes are shown in Table 4.2. The timing diagram of the original scheme and scheme *8* are displayed in Figure 4.7.

As standard setting, scheme *8* was chosen, because the afterpulsing was lowest. Here, the recharge of the diodes is done passively via the tunable quenching resistor. With all different schemes, the duration of the overall control cycle was not changed. It seems as if a longer cycle further increases photon detection probability (PDP), but no further investigations were done on this effect.

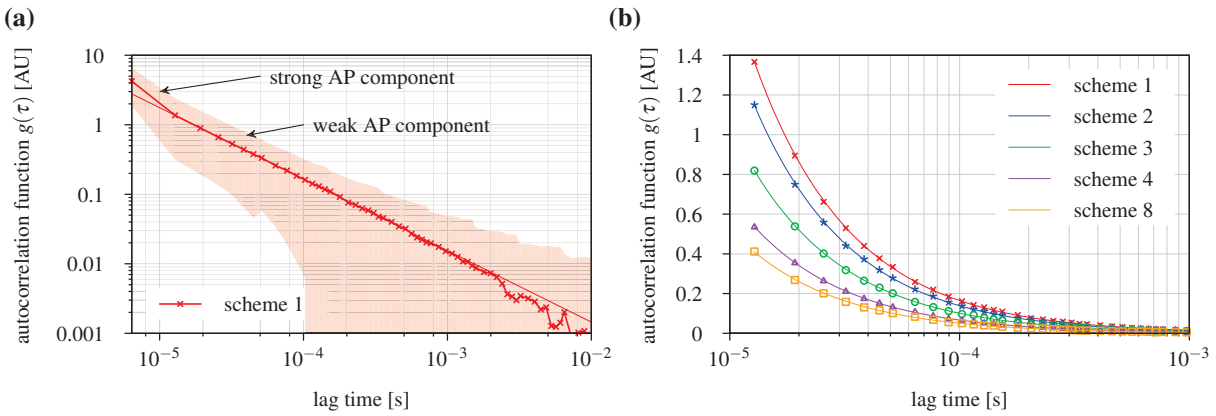


Figure 4.6.: Mean autocorrelation curve of the dark count rate (DCR) for the CHSPAD sensor without microlenses *D1*. (a) Log-log plot of the autocorrelation function including standard deviation starting with the first channel. Raw data is shown as dots, the corresponding fit as a thin line. (b) Comparison of different gating schemes. Raw data (dots) and a fit (solid line). Schemes 5,6 and 7 (not shown) are comparable to 8, but with a slightly higher amplitude. The first channel is omitted. Row data is shown as points, fits are shown as solid lines.

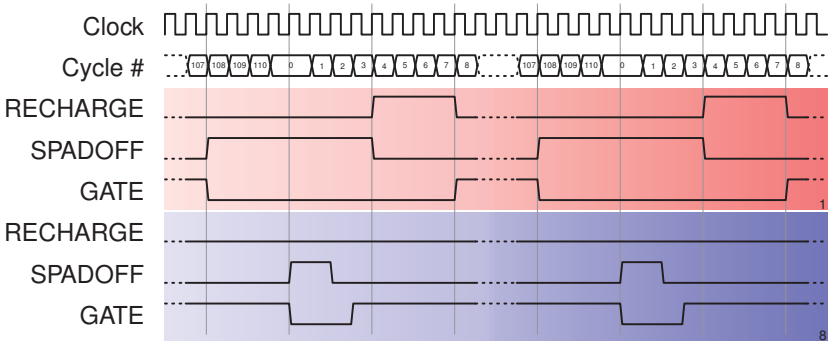


Figure 4.7.: Timing diagram of two repetitions of the control signals for CHSPAD that are common for all pixels. The original scheme is shown in red, the optimized in blue.

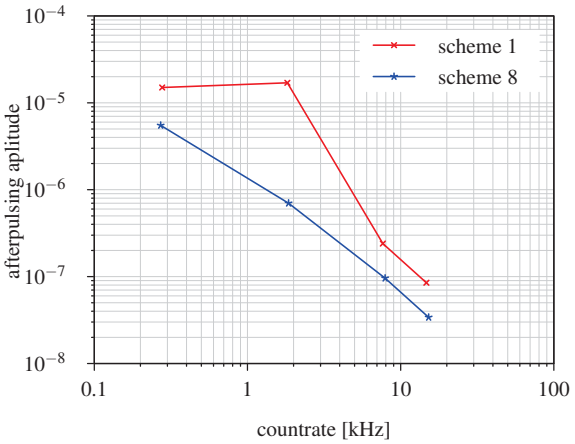


Figure 4.8.: CHSPAD: Afterpulsing amplitude as a function of the count rate. For each dataset the median value of all pixels is shown. Except for the outliers, the amplitude of afterpulsing is about 2.5 fold lower with gating scheme *8*. Additionally, the count rate is about 2 % higher, except for the dark count rate (DCR) value (leftmost dataset).

Scheme	length	SPADOFF		RECHARGE		GATE		AP amplitude [10^{-6}]
		ON	OFF	ON	OFF	ON	OFF	
1	110	106	2	2	6	6	106	1.37
2	110	106	2	3	6	6	106	1.15
3	110	106	2	4	6	6	106	0.82
4	110	106	2	5	6	6	106	0.54
8	110	0	1	1	1	1	110	0.41

Table 4.2.: Influence of the control signal scheme on the afterpulsing of the CHSPAD. The length is given in clock cycles. Signal names a shown according to Table 2.4. Values for ON and OFF represent the respective clock-cycle. Scheme 8 uses passive quenching only. The amplitude is extracted by a fit of the mean ACF estimate (*cf.*, equation (2.61)). Measurements are based on the DCR only.

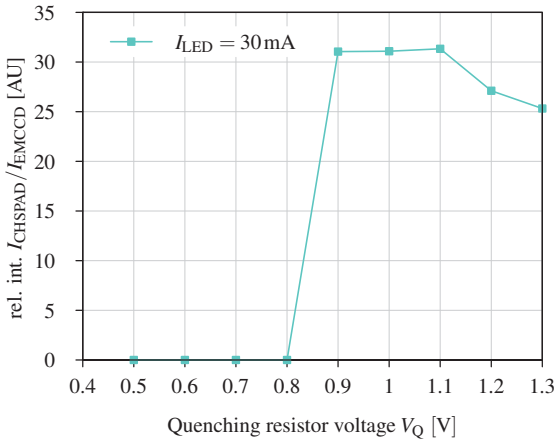


Figure 4.9.: CHSPAD: detected light intensity as a function of V_{Q}. Data is normalized to the EMCCD camera that was used in parallel (50:50 beam splitter). The measurement was done with the LED light source.

Influence of the count rate on the afterpulsing amplitude. In Figure 4.8 the influence of the count rate on the amplitude of the afterpulsing is shown. By using passive recharge, the afterpulsing is about 2.5 fold lower in comparison to the original gating scheme for almost two orders of magnitude.

Optimizing V_{Q}. To optimize the voltage applied to the tunable quenching resistor, its relation to the PDP of the CHSPAD was evaluated. The results are shown in Figure 4.9. An electron multiplying charge coupled device (EMCCD) camera was used in parallel to normalize the results for the illumination intensity. With regards to the detection efficiency, a value of $V_{\mathrm{Q}} = 0.9\,\mathrm{V} \ldots 1.1\,\mathrm{V}$ delivered best results.

4.2.5. Design of a fast readout for real-time data acquisition

A first readout chain was developed to be used for the first sample of the CHSPAD (without microlenses, start of 2013). It allows streaming the full raw data at maximum clock speed (80 MHz clock, 156 250 fps) to the host computer in real time. For this purpose, data is streamed from the *Broaddown 4* development board to a XILINX *ML605* FPGA development board [254] installed into the computer that is used for recording. The XILINX *ML605* is then used as a bus bridge between the serial link from the *Broaddown 4* and the PCIe interface of the host computer. Figure 4.10 gives an overview of the entire system. Figure 4.11 illustrates the data path including bandwidths of the different interfaces. Details are given in the following.

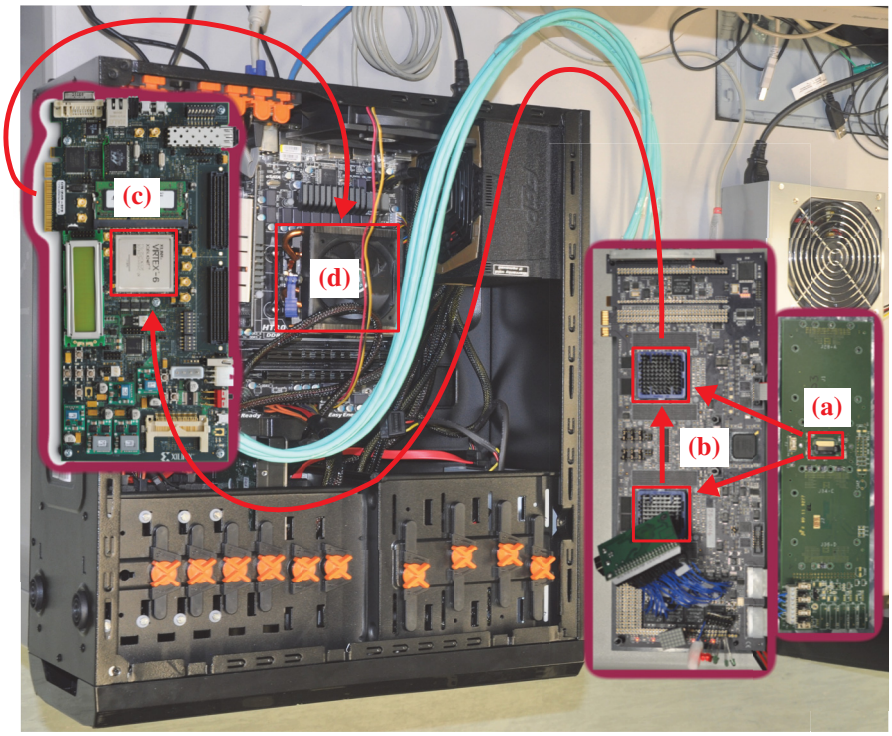


Figure 4.10.: Overview of the CHSPAD real time readout system. (a) CHSPAD on daughter board. (b) *Broaddown 4* development board with two FPGAs each connected to the CHSPAD. The daughter board is attached from the back. (c) XILINX *ML605* development board used as bus bridge to interface the PCIe bus of the host computer. (Image: XILINX) (d) CPU of the host computer. Red arrows specify the direction of dataflow.

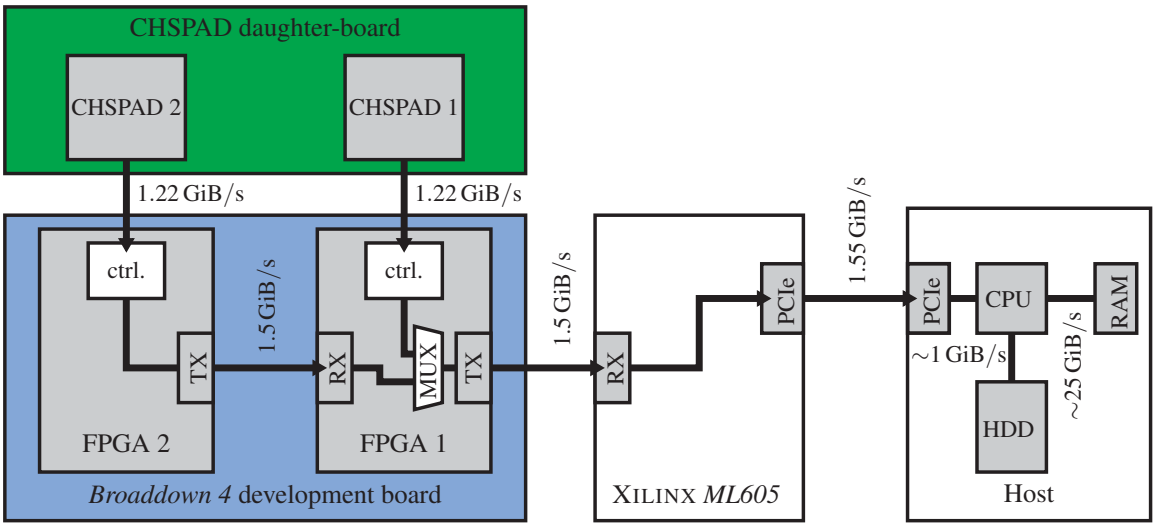


Figure 4.11.: Data path of the CHSPAD real time readout design. The first FPGA on the *Broaddown 4* board can either be used for data acquisition and transmission to the host computer, or for forwarding data from the second FPGA. Bandwidths are given per interface.

Broaddown 4 to *ML605*

The raw data rate of the CHSPAD sensor is 1.22 GB/s at 80 MHz clock speed. The development board used for readout does not feature any specific interface (*e.g.*, an optical link) that is capable of handling this data rate. Although the board is equipped with a PCIe connector, it is only capable of transferring up to 250 MiB/s due to the used bridge chip. Another option would have been a redesign of the mother or daughter board, during which an appropriate interface could have been added. But, the CHSPAD daughter board was specifically designed for the *Broaddown 4* board and a redesign was no option*. So a different solution had to be found that is able to deal with the data rate.

Another problem was the distance between the sensor and the readout computer, as well as the limited space within the microscope. The CHSPAD daughter board is directly attached to the *Broaddown 4* development board: Therefore it has to be placed directly in the optical path of the SPIM. A host computer that is used for data processing or at least for data storage needs to be placed outside the light-shielding box, without contact to the optical table to guarantee vibration isolation. Therefore, a flexible cable connection between the sensor in the microscope and an external computer, with at least 1 m length, was required.

The option that was finally chosen and fulfills both requirements is based on the "edge connector" of the *Broaddown 4* board that makes 20 differential pairs available. These were connected with the corresponding pins of the *ML605* board. The latter serves as an interface converter between the host's PCIe bus and the cable.

To interface IO-pins of the XILINX *ML605* IO-pins, a special high pin-count connector (HPC) is required. Here, the XILINX *FMC XM105 debug card* is used as a break-out to standard 2.54 mm pin-headers. The connection is formed by five *Cat 6A* cables of a length of about 1 m. One of the 19 differential pairs transmits the reference clock.

To achieve DC-balancing†, the 8b/10b line code [235] is applied to each differential signal pair. Beside that, the whole link is HAMMING encoded [80], and 6 of the 19 differential pairs transmit parity information. Each data word has an extended HAMMING distance of four instead of three with an additional parity bit (also known as *SECDED*). This scheme allows correcting single bit errors and to detect double errors. Such an error correction was especially useful if a single differential pair showed a large amount of bit errors.

As only 13 differential pairs are used to transmit sensor data, each link needs a bandwidth of 1000 Mbit or 500 MHz double data rate (DDR) to achieve the required 1.22 GiB/s. Although this is above the specification of the VIRTEX-4 serial transceivers (400 MHz reference clock frequency range, *cf.*, Ref. [245]), tests showed (see below) that the FPGA can cope with the higher clock rate.

The same type of connection was also used for the communication between the two VIRTEX 4 FPGAs on the *Broaddown 4* board. Depending on the position of the SPAD array‡, the first or the second FPGA is in control of the sensor and in some cases data has to be transferred to the other FPGA first. The high-speed clock needed for serial data transmission is generated externally by a clock generator. As the same clock source is used for both FPGAs on the *Broaddown 4* board, no additional clock signal is transmitted on this interconnect.

Bit error rates of the serial links. A single word, as transferred between the FPGAs on the *Broaddown 4* or between the *Broaddown 4* and the XILINX *ML605*, consists of $13 \cdot 8\,\text{bit} = 104\,\text{bit}$ of user data. Due to 8b/10b and hamming encoding, actually 190 bit are transmitted over the wires for every 104 bit of

*In late 2014 a new daughter board was announced, featuring a SPARTAN-6 for and a high pin-count connector as it can be found on most XILINX development boards. But it is not available yet.

†DC-balanced signals are required to prevent bit errors in capacity coupled circuits such as high-speed communication systems.

‡For tests with and without microlenses two different daughter boards were used with the SPAD array wired to different positions.

user data. For a stability test, 4.90×10^{13} words were transmitted (duration $\sim 110\,\mathrm{h}$) at 600 MHz DDR. During the test, 2.06×10^{5} single bit errors occurred in the raw data and were successfully corrected. This gives a word error probability of $p_{\mathrm{word}} = 4.20 \times 10^{-9}$ or $p_{\mathrm{bit}} = 2.21 \times 10^{-11}$ for single bits. The occurrence of an uncorrectable error (2 bits in 19 bits, $\frac{1}{10}$ of a word.) is given by

$$P(2) = \binom{19}{2} \cdot (p_{\mathrm{bit}})^2 \cdot (1 - p_{\mathrm{bit}})^{19-2} = 8.35 \times 10^{-20}. \tag{4.1}$$

At the given data rate, this is likely to happen less than once per year. Using both encodings in combination should enable the detection and correction of further errors. In addition, even if a few bits flip within a dataset, the impact on the correlation analysis is negligible, due to the overall large amount of data. At a lower clock rate of 500 MHz DDR (this was used in the final design) almost no errors occurred in the raw data.

ML605 to host computer

The *ML605* development board is used as an interface converter between the serial link from the *Broaddown 4* and the PCIe bus of the host computer. Hardware and software used for PCIe readout were provided by HEIKO ENGEL (Frankfurt University, Germany).

Arriving data is buffered in an on board DDR-3 RAM and then transported to the host computer via a PCIe v2.0 x4 connection. Of the theoretically peak bandwidth of 2000 MB/s, approximately 1554 MB/s were achieved. A full simulation of the hardware and software predicted 1580 MB/s.

4.2.6. Data compression

The enormous data rate generated by the CHSPAD demands for high-speed interfaces on the hardware platform, *Broaddown 4*, that were not easy to implement. Although high-speed interfaces could be improvised by combining the *Broaddown 4* with the XILINX *ML605* as bus bridge to PCIe and by operating the VIRTEX 4 transceivers above their specifications, a compression of the amount of data to be transferred is desirable. For that purpose, two possibilities were examined: One is to reduce the data rate by selecting a subregion of the active area, but this neutralizes the advantage of the high resolution of the sensor. Another possibility is lossless data compression based on entropy encoding, which is discussed in the following.

Lossless data compression

Lossless data compression is an algorithm that allows the perfect reconstruction of the original data from the compressed data. The input alphabet A comprising n symbols that describe the input data,

$$A = \{a_1, a_2, \ldots, a_n\}, \tag{4.2}$$

and a set of symbol weights P for each symbol of the alphabet, which is usually proportional to the probability,

$$P = \{p_1, p_2, \ldots, p_n\}, \tag{4.3}$$

are mapped by the compression algorithm to a set of code words C, the output:

$$C(A,P) = \{c_1, c_2, \ldots, c_n\}. \tag{4.4}$$

The compression is efficient, if the length L of the encoded input data is less than the original input:

$$L(C) = \sum_{i=1}^{n} w_i \cdot \mathrm{length}(c_i) \leq L(T), \quad \text{for any code } T(A,P) \tag{4.5}$$

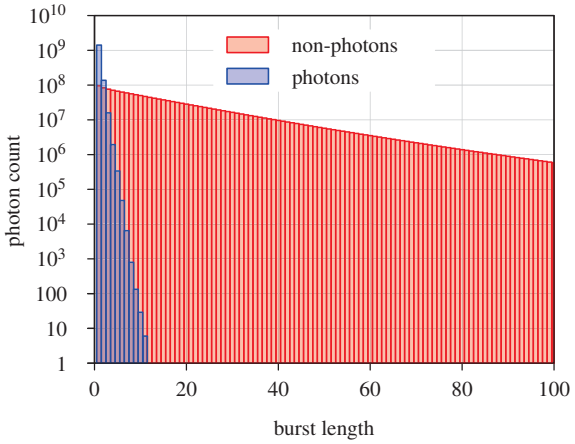


Figure 4.12.: Histogram of photon and non-photon bursts used for entropy encoding of SPAD data. The ROI was 512×4 pixel wide, and 131 072 frames were taken into account. The length of the longest non-photon burst was approximately 400. Bursts were counted by walking through the entire dataset row-major order.

If the raw and encoded data are a set of binary codewords, their length is the number of required bits. The code C must be prefix-free and obey Kraft's inequality [116] to be uniquely decodable.

The challenging tasks is to find a symbol alphabet that describes the input data and allows for efficient compression. For low light applications, such as FCS, only few photons are expected with long pauses in between. Thus, the symbol alphabets described in the following are based on the run length of photon bursts.

Burst-length encoding

In FCS, with molecules entering and leaving the focal volume, per pixel a time series of rare photon events and long pauses in between is expected. This is supported by Figure 4.12, which shows a typical histogram of photon and non-photon bursts for a *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads measurement (see section 2.2.4 for details on the dye and chapter 6 for details on the measurement procedure). Thus, a compression scheme based on the length of the bursts seemed to be a valid choice. In a next step, the symbol alphabet for the input data has to be found.

Constructing the input symbol alphabet. In the following three different symbol alphabets A were tested. As the matching process of the raw data onto the symbols should be done in hardware, the alphabet need to be as simple as possible. First, the most naïve set comprises different bursts of either N_1 photons or N_0 non-photons:

$$A_{\text{naïve}} = \{0, 00, 000, \ldots, 0\ldots0, 1, 11, 111, \ldots, 1\ldots1\} = \{0^a | 0 < a \leq N_0\} \cup \{1^a | 0 < a \leq N_1\}\,. \tag{4.6}$$

Secondly, a two dimensional alphabet comprising burst of non-photons and photons with varying sizes combined into a single symbol was tested:

$$A_{\text{two-dimensional}} = \{01, 001, 011, 0001, 0011, 0111, \ldots\} = \left\{0^a 1^b | 0 < a \leq N_0, 0 < b \leq N_1\right\}. \tag{4.7}$$

Finally, an alphabet consisting of photon-terminated non-photon bursts was tested:

$$A_{\text{photon-terminated}} = \{1, 01, 001, \ldots, 0\ldots1\} = \{0^a 1 | 0 < a \leq N_0\}\,. \tag{4.8}$$

Constructing the codewords. The loss-less data compression applied to the raw data is based on entropy encoding. The HUFFMAN algorithm was used to construct an optimal prefix-free code C [94]. For every data set an individual HUFFMAN tree was constructed. Further optimizations can be done on the input alphabet A to better match the statistical nature of the raw data and to improve the compression factor. The entire process is similar to the so called *modified* HUFFMAN *coding* used for encoding monochrome images in fax machines [221].

Results. Figure 4.13 shows the performance of the symbol alphabets described above in relation to the maximum run-length of an input symbol. A symbol length equal or above 128 seems to be optimal for efficient coding. Typically, compression is done for each pixel independently (solid line). For a homogeneous sample, the behavior of all pixels should be comparable, but a dedicated compression core (if implemented in hardware) is required for every single pixel; 128×512 in total. A more hardware efficient approach is to consider the entire raw data as a row-major stream (dashed line), which only requires a single compression core. Figure 4.14 shows the result of the evaluation of measurements of enhanced green fluorescent protein (eGFP) in HeLa cells (see section 6.9.3 for details on the measurement). Although the raw data has an implicit pattern due to the inherent structure of the cell, a significant influence of this pattern on the coding efficiency in comparison to homogeneous samples could not be detected. In comparison to an output of coordinates of photon events, a significantly better compression factor is reached.

Summary

Compression based on entropy encoding seems to be a promising method to reduce the high data rate of the CHSPAD sensor. The measurements showed that there is no significant difference, whether the data is encoded in time for each pixel individually or spatially line-by-line ('row-major'). The latter should have a much simpler implementation in hardware. Since the coding efficiency strongly depends on the count rate from the sample, entropy encoding is especially well-suited for low light single-photon measurements. For very dim samples, a more than 20 fold reduction in the data rate was achieved. Even if the count rate is increased beyond 10 kHz, *e.g.* with a more sensitive sensor, one could still compress the raw data stream.

4.2.7. Computer storage

For further analysis, data must not only be transferred to a host computer, but also to a long-term storage, preferably low-cost hard disks or network attached storage. This has to be done in real time in parallel to the ongoing data acquisition or time-shifted from the host's RAM. The latter typically limits the overall measurement time or measurement repetition rate (*e.g.*, 27 s for 32 GiB). The bandwidth requirements for real-time storage are slightly above the raw data rate of 1.2 GiB/s, due to the overhead of the file system. This exceeds the continuous data rate of most available conventional (rotating magnetic disk) hard disks and flash based solid state disks (SSDs). Such devices typically offer a data rate in the range of 500 MB/s. But by combining multiple of those disks, *i.e.*, in a redundant array of inexpensive disks (RAID) level 0 configuration*, the desired data rate can be achieved.

Current consumer mainboards feature about four to eight SATA-3 (6 GB/s) ports. As one is typically used for the system disk, the remaining ports can be used with a host-based (*i.e.*, software) RAID solution†. The SATA connectivity is typically located in the southbridge part of the computer's chipset,

*Strictly speaking, a striping configuration is no RAID, as no redundancy nor fault tolerance is given. It is just an array of independent disks.

†Nowadays this functionality is offered by the operating system.

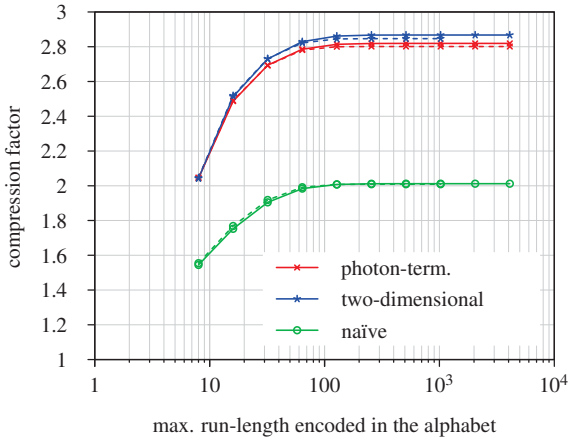


Figure 4.13.: Influence of the size of the histogram on the compression factor. Solid lines represents spatial (row-wise) encoding, dashed lines temporal (per pixel) encoding. The naïve implementation using two histograms for photon and non-photon burst is shown in green (see equation (4.6)). The 'photon-terminated' implementation (see equation (4.8)) is shown in red and the two-dimensional approach in blue (see equation (4.7)).

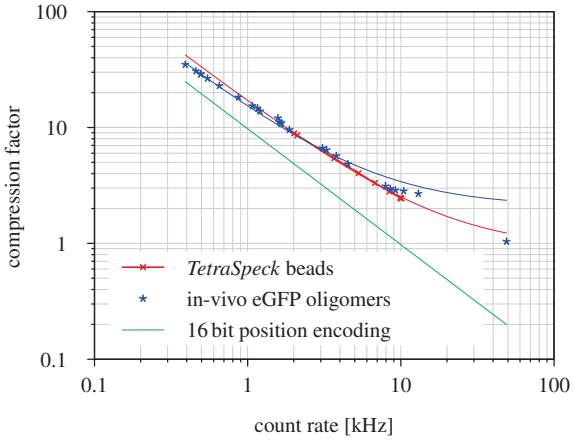


Figure 4.14.: Efficiency of the entropy coding in relation to the sample count rate. An excitation laser intensity series of *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads is shown in red. The compression efficiency of samples of eGFP oligomers in HeLa cells are shown in blue. The symbol alphabet was 'photon terminated' (see equation (4.8)). HeLa cells were recorded with a 128 pixel wide ROI best fitting the cell, *cf.*, Figure 6.23. Hyperbolic fits of the data are shown as a thin line. Laser power was $80\,\mathrm{W/cm^2}$. A compression based on 16 bit position encoding is shown in green.

which usually implements the slower capabilities of the motherboard. In todays computers, the south-bridge is connected to the CPU by a PCIe v2.0 x4 like interface (*i.e.*, AMD *A-Link Express III* or INTEL *Direct Media Interface 2.0*). This limits the theoretical peak performance of 2 GB/s. Additionally, this link is shared with any other transfer from or to the chipset.

Tests showed, that the continuous data rate of 1.2 GiB/s cannot be achieved, even if four SAMSUNG *840 PRO* are used in a software RAID 0 configuration. This problem might be overcome by using more recent SSDs that are using a PCIe interface, and can therefore communicate with the CPU directly (*e.g.*, *M.2* form factor).

4.3. Characterization of the CHSPAD sensor

Several characterization measurements of CHSPAD were done which are described in the following sections. For these measurements, the sensors were integrated into the SPIM setup (see Figure 2.20 for details on the microscope set-up). In total, two different chips were utilized: A sensor with microlenses *C21* and a sensor without microlenses *D1*, which was mostly used for comparison. The sensor with microlenses was hand-selected by the designer of the chip and showed above average sensitivity. The sensor without microlenses was only used for comparison and showed some defects (short circuit on V_Q, large defects on the sensor area, and significantly higher power consumption). If not stated differently, all measurements were performed using a sensor with microlenses.

4.3.1. Sensor alignment procedure

Prior to any measurement, the sensors have to be aligned. The SPIM setup was already aligned for the EMCCD camera, thus the following steps are specific to the CHSPAD: First, the position of the CHSPAD is adjusted to the focal plane of the tube lens. If the camera is used in parallel, both sensors need to be aligned in a way to have an overlapping field of view, which is shown in Figure 4.15. This is done by moving the x-y-stage of the sensor's mounting. A sharp metal tip, which is mounted in the water filled sample chamber, is used as a reference sample.

As a next step, a possible misalignment of the microlenses is corrected: By adjusting the sensor's pan and tilt, the intensity maximum is centered (see section 4.3.7 for details). In case of a sensor without microlenses, no further actions are required and the chip is adjusted perpendicular to the optical axis.

4.3.2. Signal to noise ratio

An important characteristic of the light detection process is the amount of noise that is added to the signal. For an ideal detector without further sources of noise, the probability to detect photons follows POISSONIAN statistics. Further sources of noise are thermal noise due to random creation of electron-hole-pairs and readout noise of the analog to digital conversion process. A quantification of the quality of a signal is the signal to noise ratio (SNR) which is defined as the ratio of the signal power P_s to the noise power P_n:

$$\mathrm{SNR} = \frac{P_s}{P_n}, \tag{4.9}$$

with the signal s and the noise n. Assuming a detection process like the following:

$$X = s + n \tag{4.10}$$

with X being the detected intensity, s the constant intensity signal and n a random variable modeling all different kinds of noise with an expected value equal to zero. The SNR then equals

$$\mathrm{SNR}(X) = \frac{s^2}{\sigma_n^2}. \tag{4.11}$$

(a) EMCCD **(b)** CHSPAD

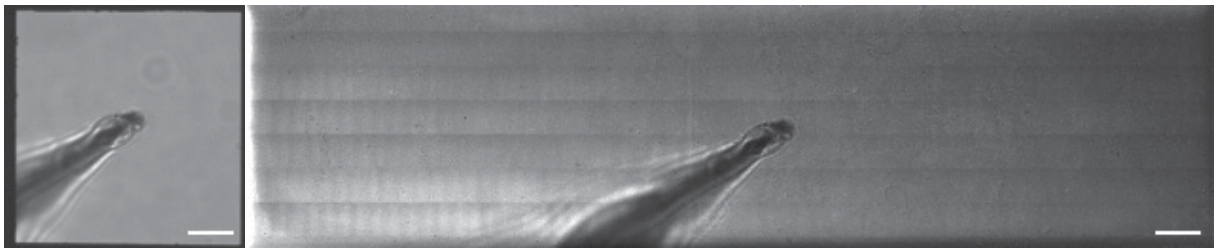


Figure 4.15.: SPIM alignment. Parallel adjustment of (a) EMCCD camera and (b) CHSPAD using a small metal tip. Height of both images: 128 pixels, width: 128 and 512 pixels. Ruler: 9.6 µm in the image plane (576 µm on the sensor). Images are brightness and contrast enhanced.

For an ideal detector where the detection probability of a photon follows the rules of POISSONIAN statistics, this can be written as

$$\mathrm{SNR}(X)_{\mathrm{ideal}} = \frac{s^2}{\sqrt{s^2}} = s. \tag{4.12}$$

Measurement procedure

For each sensor, a series of images is taken for multiple LED light intensity settings. From the images, the mean value and the variance were calculated. In case of the EMCCD, 1000 frames were taken at 10 ms integration time. For the CHSPAD array, 25 frames with 16 bit resolution were taken at 419 ms integration time. Prior to calculation, the background is subtracted from the raw images. To verify the position independence of the SNR in case of the CHSPAD, the visible area was divided into tiles of 32×32 pixels. Every tile was handled as an independent measurement. For the EMCCD camera the intensity values are converted into photoelectrons by a fitting procedure (*cf.*, [168]).

For the measurement the typical SPIM setup was used without sample chamber and with the LED as light source. The currents of the LED illumination ranged from 0 mA to 1000 mA. To access even lower light intensities, neutral density filters (OD 1 and OD 2) were mounted in front of the light source.

Results

Figure 4.16 shows the result of an SNR measurement of the CHSPAD (red) and the *iXon X3 860* EMCCD. In case of the EMCCD camera, different EM-gain settings were used (orange, pink, green and blue). For comparison, the characteristic SNR of an ideal photon detector (POISSONIAN noise) is shown (black). The respective values are plotted against the detected count rate. A raise in the PDP would result in a right shift of the curve along the straight line of the ideal sensor.

The plot shows that the CHSPAD can be treated as a ideal sensor for count rates above 1 kHz. Below that value, the DCR seems to have a stronger influence (see next section). For values above 10 kHz, the CHSPAD's SNR seemed to outperform the ideal sensor. This was an artifact, since in this regime the probability for double-photons becomes significant and therefore several photons are not detected, which limits the SNR over-proportionally. Theoretically, the upper limit for the count rate is 156 kHz which cannot be reached for larger areas of the sensor due to electrical restrictions*. The EMCCD camera, on the other hand, with its stochastic amplification in the EM-gain register suffers from an additional noise component (excess noise).

*Each detected photon creates an avalanche of electrons. The higher the overall intensity, the more current is drawn by the sensor. To prevent the sensor from damage, the total current is limited to 600 mA.

4.3.3. CHSPAD dark count rate

A key parameter in defining the detector noise is the so called dark count rate (DCR), which is the rate of detected pulses when the sensor is in complete darkness. As explained in section 2.1.2, these are caused by thermally created electron-hole-pairs in the sensor substrate. The influence of the excess bias V_{ex} (see next section) in the DCR is shown in Figure 4.17a. A histogram of the DCR at a typical value of $V_{\mathrm{OP}} = 24\,\mathrm{V}$ for different sensors with and without microlenses is given in Figure 4.17b.

The higher DCR of the sensor without microlenses (*D1*) most likely stemmed from inherent defects of the die (*cf.*, Figure 4.26). In case of the sensor *C21*, the obtained value of 100 Hz to 200 Hz was slightly better than the reference (211 Hz for $V_{\mathrm{ex}} = 3.5\,\mathrm{V}$ [33]).

4.3.4. Breakdown voltage

When operated in Geiger-mode, the breakdown voltage V_{break} is the bias voltage over the sensor, above which avalanches can occur. A higher voltage bias above breakdown, the excess bias V_{ex}, increases the gain (and DCR) of the sensor. The sum of both, the operating voltage $V_{\mathrm{OP}} = V_{\mathrm{break}} + V_{\mathrm{ex}}$, is limited by the chip production process ('high-voltage CMOS'-process for the CHSPAD). The excess bias is typically used to compare properties of different sensors, as for example the light sensitivity. The breakdown voltage is measured as described in the following.

Measuring a SPADs breakdown voltage. If the diode is individually accessible, the breakdown voltage is measured by observing the *I-V* of the diode. The voltage is reached, when a bifurcation in the characteristic occurs (photon and non-photon states). If the diode is not directly accessible, which is especially true for digital SPAD-arrays like the CHSPAD, a different approach is used: For different operating voltages V_{OP} a series of single frames were acquired. A single pixel was considered to be active, when at least one photon was counted. This can either be measured in the dark, so that only the DCR plays a role, or under light conditions. By counting the number of active pixel for every V_{OP}, a cumulative distribution function of V_{break} is obtained. The mean value can then be determined by a fit.

Results

Figure 4.18 shows the histogram of the breakdown voltages across all pixels for both SPAD arrays. The mean value was obtained from a GAUSSIAN fit. The values slightly diverged for different settings of V_{Q}. A color-coded image of the breakdown voltage on the whole chip is shown in Figure 4.19. The map shows that the breakdown voltage is not evenly distributed but shows some large-scale structures, which may stem from substrate imperfections.

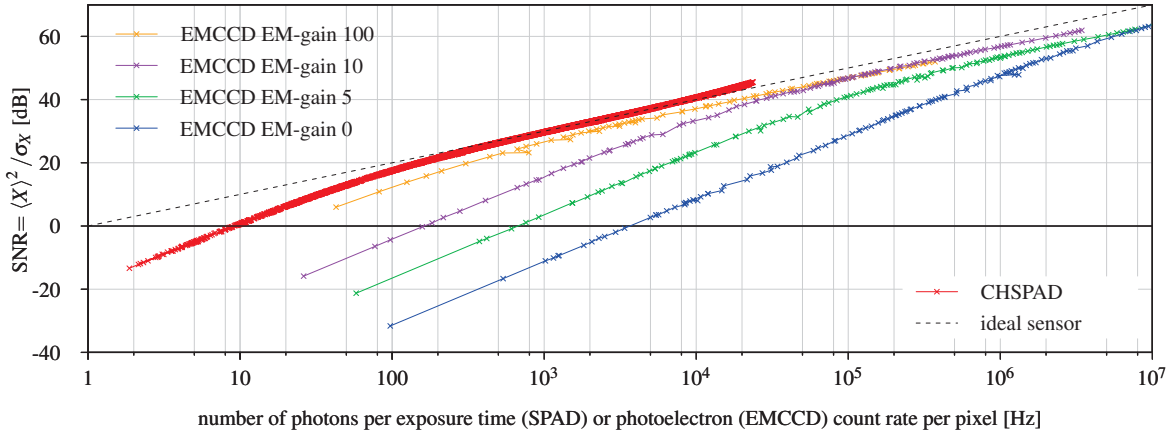


Figure 4.16.: Signal to noise ratio of the CHSPAD (red) and the EMCCD camera plotted against the detected count rate. Results for the CHSPAD are shown in red. For comparison, the SNR of an ideal sensor (POISSONIAN photon detector, black line) is shown. Solid lines are bezier interpolations of the raw data. The different sensors were illuminated by the LED light source with I_{LED} ranging from 0 mA to 1000 mA. To access even lower light intensities, neutral density filters were attached to the light source. The single frame integration time of the CHSPAD were 419 ms and for the EMCCD 10 ms. EMCCD data was provided by JAN W. KRIEGER.

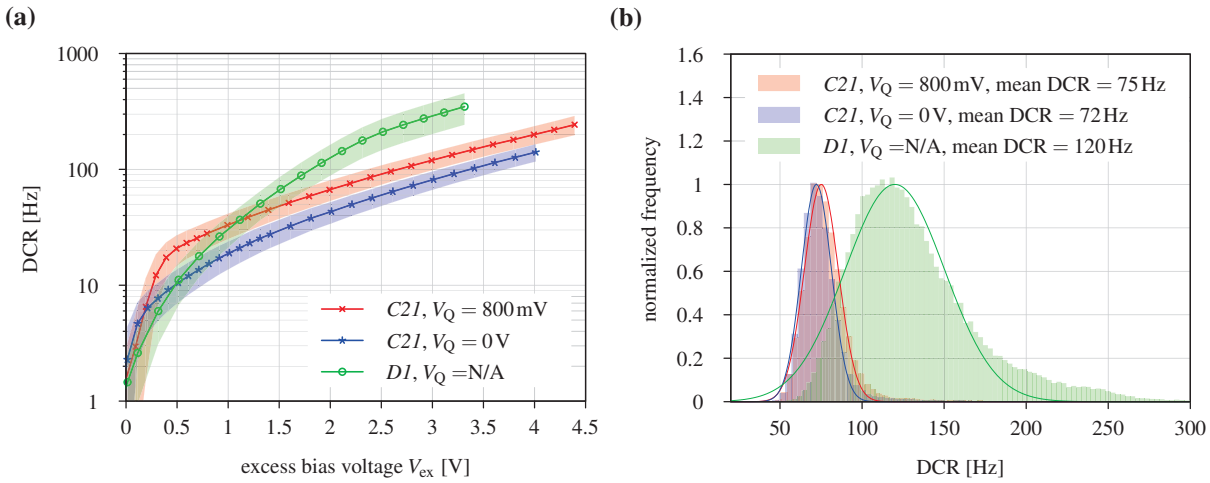


Figure 4.17.: Measurement of the dark count rate of the CHSPAD sensor. (a) Dependence of the DCR on the excess bias voltage V_{ex}. Semi-transparent areas indicate standard deviations. (b) histogram of the DCR for all pixels. $V_{\mathrm{OP}} = 24$ V.

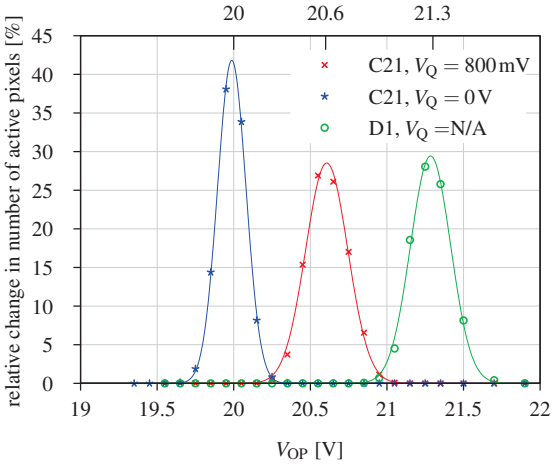


Figure 4.18.: CHSPAD: Determination of the breakdown voltage V_{break}. On the ordinate, the change in the number of active pixels in a DCR measurement is plotted. Solid lines indicate GAUSSIAN fits, the mean values from fit are shown above. Typically 5 frames with 16 bit resolution were recorded. Pixels with more than a single count were considered to be active.

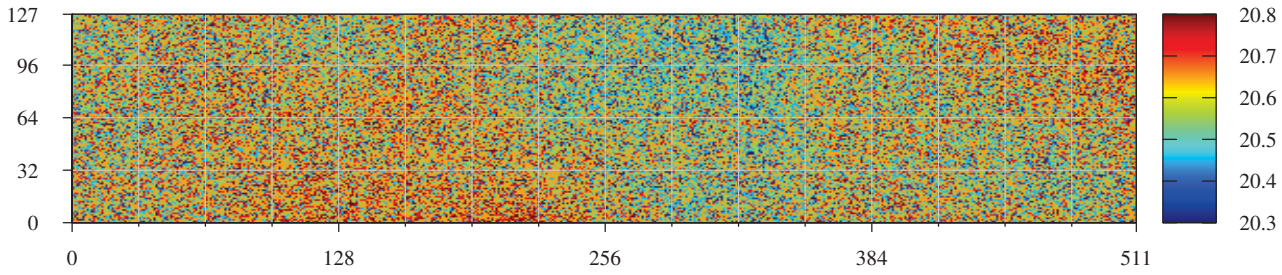


Figure 4.19.: Distribution of V_{break} across the entire CHSPAD sensor *C21*. A pixel is considered as active when more than one photon is detected in a series of 5 images with 16 bit resolution each. The corresponding distribution for all pixels is shown in Figure 4.18. Data is based on a DCR measurement. $V_Q = 800\,mV$.

4.3.5. Detection efficiency

By increasing the excess bias, the detection efficiency is increased. This effect is shown in Figures 4.20 and 4.21.The first, Figure 4.20, shows the detected number of photons at a constant illumination for different values of V_{ex}. The second, Figure 4.20, shows the linearity of the detection for different values of V_{ex} by comparing the detected number of photons with the intensity measured with the EMCCD camera in parallel.

4.3.6. Final test of the hardware-software co-design for correlation analysis

To ensure the functional correctness of all hardware and software components involved in the data acquisition and evaluation process, a homogeneous sample with a known timing characteristic was measured and evaluated. After fitting an appropriate model function to the estimates of the ACFs of every pixel, the obtained parameters should have a narrow distribution if all pixels behave similar. As sample, a white LED light source was used in a frequency-modulated mode. The frequency was set to $f_{\mathrm{LED}} \sim 1\,\mathrm{kHz}$. Adequate optics ensured that the entire sensor was illuminated uniformly. The oscillation model (*cf.* section 2.4.3) was used as fit model for the ACCs of all pixels.

The duration of the measurement was 3.35 s, and full frames were acquired. An average count rate of $(40 \pm 1)\,\mathrm{kHz}$ was observed. Figure 4.22 shows the distribution of the frequency parameter obtained from the model fits. A GAUSSIAN fit of the histogram yielded a mean frequency of $(1042.43 \pm 0.06)\,\mathrm{Hz}$. The divergence between the set frequency and the mean value obtained from fit ($\approx 4\,\%$) might stem from the frequency generator from which the LED frequency was derived (micro-controller based, 16 MHz clock, see Ref. [119]). The very small width of the distribution of the frequencies obtained from model fits of each pixel showed that the pixels behave similarly and that the entire system was working as expected.

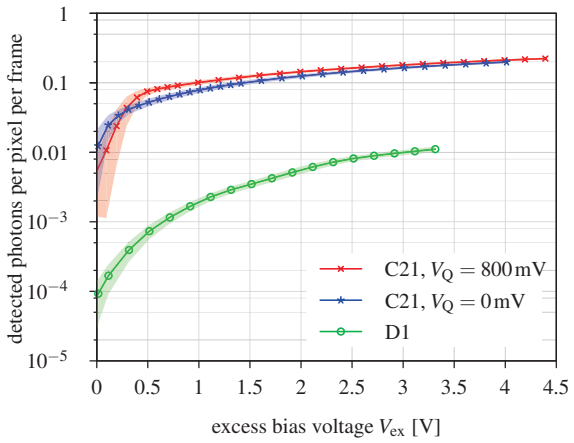


Figure 4.20.: Detection efficiency of the CHSPAD in relation to excess bias voltage V_{ex}. Standard deviation is shown as semi-transparent area. To account for the inhomogeneous illumination of the sensor, the brightest 96×96 pixel subregion was selected. The LED was used as light-source. $I_{\mathrm{LED}} = 30\,\mathrm{mA}$.

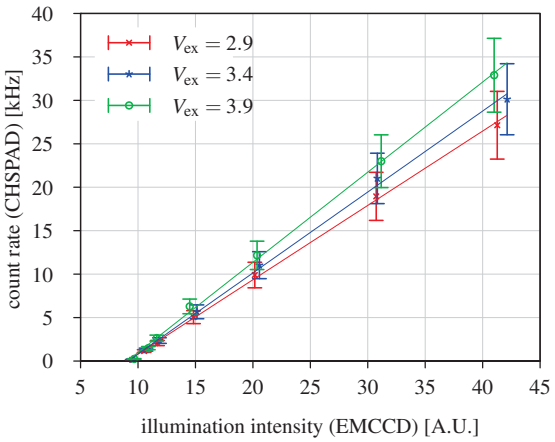


Figure 4.21.: Effect of the excess bias voltage V_{ex} on the detection efficiency of the CHSPAD with microlenses *C21*. The count rate of the CHSPAD was measured in the brightest 96×96 pixel subregion. As a reference, the EMCCD camera was used. Thin lines are linear fits, error-bars indicate standard deviation. Defective pixels were not taken into account. $V_Q = 1.10\,\mathrm{V}$.

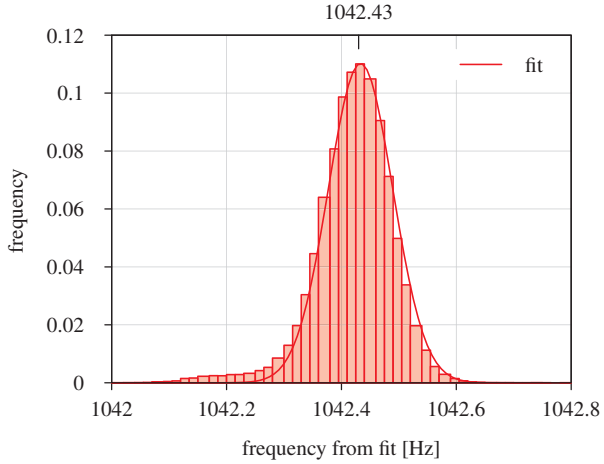


Figure 4.22.: Histogram of the frequency distribution obtained from a fit of the ACC of each pixel for a frequency-modulated uniform illumination of the sensor. A blinking LED was used as light source, which had a frequency of $f_{LED} \sim 1\,\mathrm{kHz}$. For model fitting of the ACCs, the sine oscillation model was used (section 2.4.3). The measurement duration was 3.35 s. Only non-defective pixels of the sensor were taken into account.

4.3.7. Effect of microlenses

The overall low PDP due to a low fill-factor in cameras can be overcome by using micros-lenses. Therefore a small lens that focuses the light onto the active area, a so called microlens, is mounted in front of each single pixel. Figure 4.23 shows an image of the microlenses attached to the CHSPAD (sensor *C21*). Simulations for the CHSPAD predicted an up to ten-fold light concentration [169].

Being an additional component in the light-path of the microscope, microlenses may have a significant influence on the detection. This is assessed in the following.

Influence of microlenses on the optical system

The microlenses that were placed on top of each single SPAD were not aberration corrected, but were designed for parallel incident light only. Figure 4.24a shows the effect of the optical system: If, as in case of a microscope set-up, a tube-lens is used to focus down the light onto the sensor, the focal spot of the microlenses is not necessarily overlapping with the active area inside the pixel any more. This effect gets more severe with increasing distance of the center of the field of view. Figure 4.24b shows a possible correction scheme for this telecentric error by a local displacement of the microlenses: With increasing distance, the position of microlenses is displaced towards the center.

The non-optimal placement of the microlenses lead to a visible drop of the intensity towards the border of the field of view, which is shown in Figure 4.25a. Compared to the center, roughly a factor of two in intensity is lost. By using parallel light for illumination, as shown in Figure 4.25b, a constant signal across all pixels is measured as expected. If a sensor without microlenses was used, which is shown in Figure 4.26, the drop in intensity towards the edges could not be detected.

Concentration factor of the microlenses

An important property of microlenses is their ability to focus down the light onto the active area. This can be quantified by the so called concentration factor $\mathrm{CF}_{\mu\mathrm{L}}$, which represents the gain in intensity by microlenses. In general, it is not possible to test chips prior to the application of the microlenses, or to remove the microlenses, once they were applied. Therefore, chips with and without microlenses have to be compared to estimate $\mathrm{CF}_{\mu\mathrm{L}}$. This is not very precise as the properties of the sensors vary between chips, especially for the prototypes used in this work. And even the application process of the microlenses or the microlenses themselves may change the characteristics of a sensor. Still, an effort was made to get such an estimate.

The concentration factor $\mathrm{CF}_{\mu\mathrm{L}}$ of the microlenses was determined for various excess voltages $V_{\mathrm{ex}} = V_{\mathrm{OP}} - V_{\mathrm{break}}$ and different light intensities of the LED, that were set by its current I_{LED}. The evaluation is based on the light intensity I detected with two different sensors. It is calculated as follows:

$$\mathrm{CF}_{\mu\mathrm{L}}(V_{\mathrm{ex}}, I_{\mathrm{LED}}) = \frac{I_{\mathrm{CHSPAD\ C21,\ rel}}(V_{\mathrm{ex}}, I_{\mathrm{LED}})}{I_{\mathrm{CHSPAD\ D1,\ rel}}(V_{\mathrm{ex}}, I_{\mathrm{LED}})}, \tag{4.13}$$

with the relative light intensity I_{rel}, which is dark count corrected and normalized to a parallel measurement with the EMCCD camera* (50 : 50 beam splitter) to compensate for any variations in the LED intensity:

$$I_{\mathrm{CHSPAD,\ rel}}(V_{\mathrm{ex}}, I_{\mathrm{LED}}) = \frac{I_{\mathrm{CHSPAD}}(V_{\mathrm{ex}}, I_{\mathrm{LED}}) - I_{\mathrm{CHSPAD}}(V_{\mathrm{ex}}, I_{\mathrm{LED}} = 0\,\mathrm{mA})}{I_{\mathrm{EMCCD}}(I_{\mathrm{LED}}) - I_{\mathrm{EMCCD}}(I_{\mathrm{LED}} = 0\,\mathrm{mA})}. \tag{4.14}$$

To account for image artifacts, only the inner 96×96 pixels of the EMCCD camera were taken into account. In case of the CHSPAD with microlenses the brightest 128×128 pixels were selected automatically which were then clipped to 96×96. This reduced the influence of the misplaced microlenses

*As shown in Figure C.3, the camera can be treated as linear in the evaluated range.

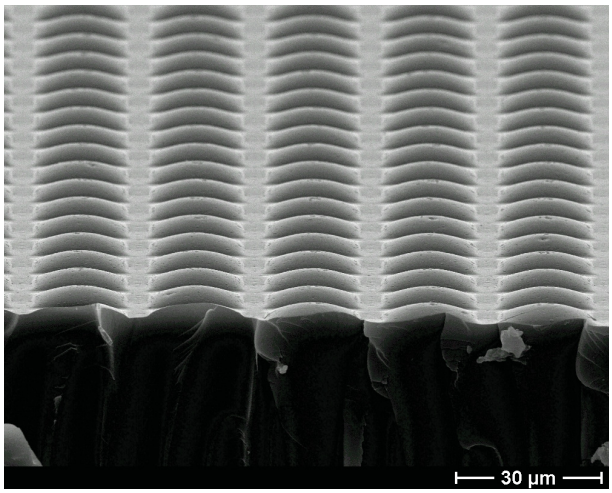


Figure 4.23.: Microscopic image of the microlenses attached to the CHSPAD. Provided by C. BRUSCHINI (EPFL, Switzerland).

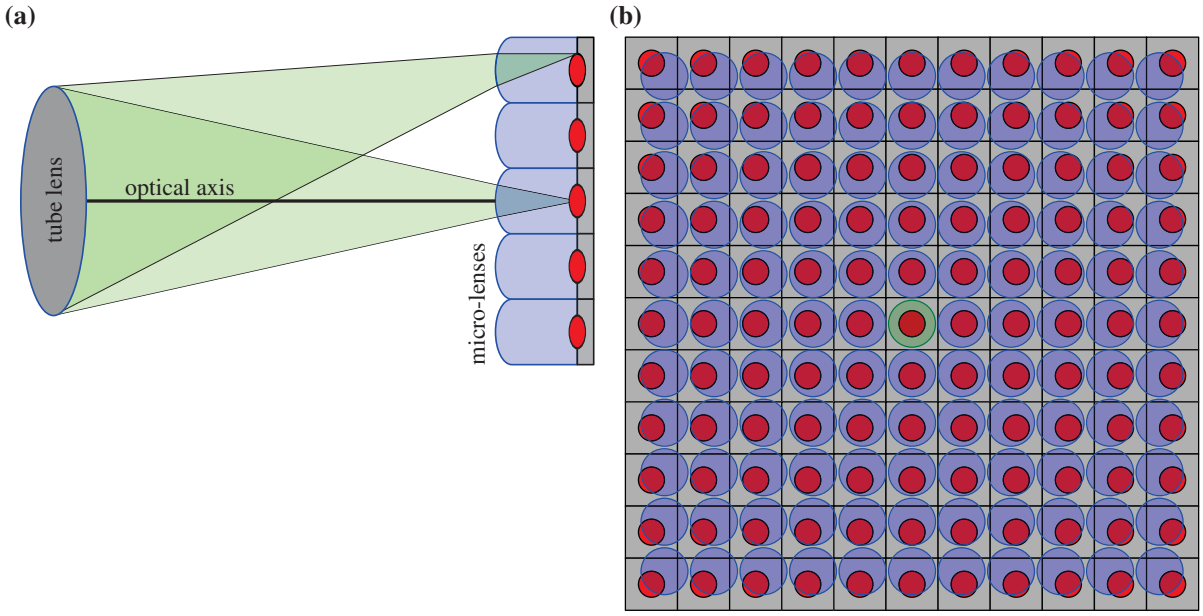


Figure 4.24.: Influence of the optical system on microlenses attached to a sensor. Pixels are depicted as gray boxes, the active area is shown in red. (a) shows the propagation of a light beam (green) through a tube-lens and microlenses for the central pixel and a pixel further outside. (b) Microlenses (blue) are displaced to account for the effect shown in (a). The displacement increases with the distance to the center (green).

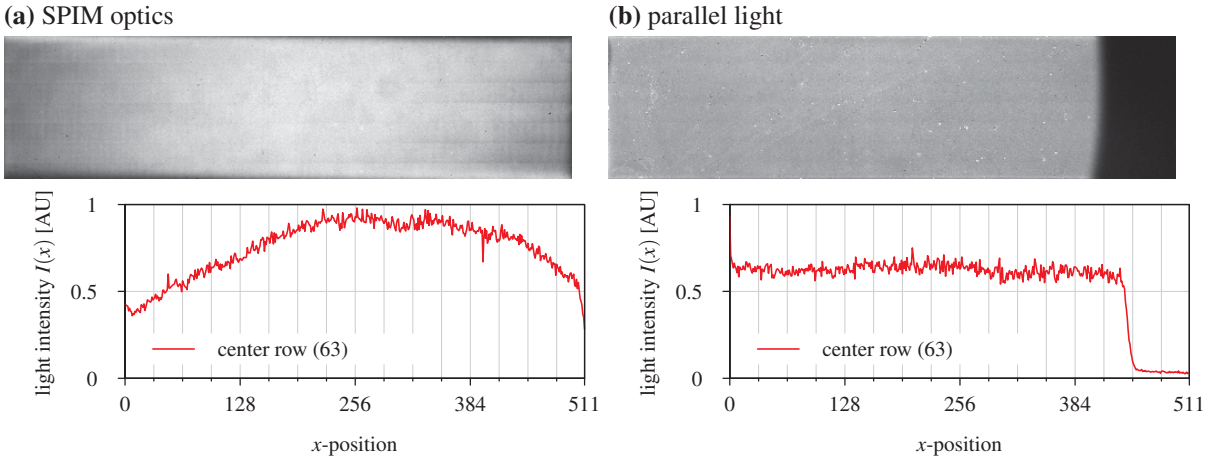


Figure 4.25.: Influence of the SPIM optics on the CHSPAD with microlenses *C21*. (a) CHSPAD mounted in the SPIM setup behind $f_{TL} = 200\,\text{mm}$ tube lens. Intensity drops about 50 % to the edges. (b) CHSPAD illuminated with parallel light. Outer right part of the sensor was not illuminated. A Gaussian fit of the histogram of intensity-values gives a variance of $\sigma = 6\,\%$). Which is mainly due to imperfections of the light source. Below the images, the progression of the light intensity for the center row is shown. For both images, dead pixels are directly removed in hardware. Remaining dark and bright spots on the left image are mainly due to dust on the sensor.

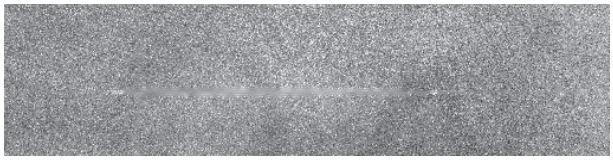


Figure 4.26.: Influence of the SPIM optics on the CHSPAD without microlenses *D1*. The image is much noisier due to the significantly reduced PDP. In the center area a large defect is visible. The image is contrast enhanced.

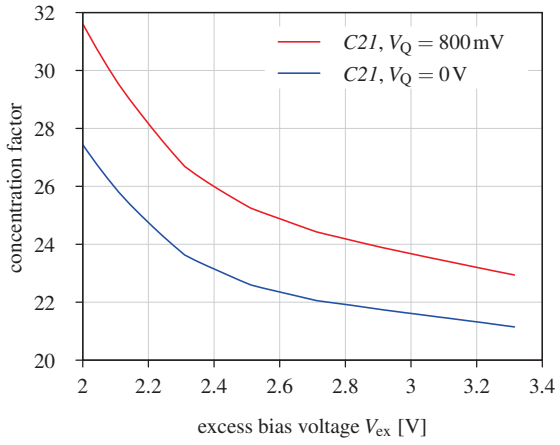


Figure 4.27.: Concentration factor of microlenses in relation to the excess bias for the CHSPAD sensor *C21*. Sensor *D1* was used as a reference. The concentration factor is shown for two different settings of V_Q. Raw data was linearly interpolated and corrected for the DCR and normalized to the count rate of the EMCCD camera. The LED light source was used for illumination; $I_{LED} = 30\,\text{mA}$. Defective pixels were excluded from evaluations.

and led to an estimate of the maximum possible gain. The sensor without microlenses was used almost completely (512×96 pixels). All measurements were normalized to their respective breakdown voltage V_{break} (*cf.*, section 4.3.4).

The alignment of the sensor was done according to section 4.3.1. As for detection a water immersion objective was used, the sample chamber was filled with purified water. Further details on the alignment of the setup are described in Ref. [119].

Figure 4.27 shows the measured gain in intensity of CHSPAD *C21* over *D1* in relation to the breakdown voltage. For excess bias voltages above 3 V a gain in intensity of approximately 22 was obtained. For $V_{\text{ex}} < 2.5\,\text{V}$ unrealistically high gain factors were measured due to nonlinearities of the CHSPAD sensors. Throughout this thesis, measurements were performed at $V_{\text{OP}} = 24\,\text{V}$ which is approximately 3 V above V_{break} and still suitable for the electronics of the sensor regarding degradation. Although the overall gain by the microlenses decreases for higher V_{ex}, the PDP still increases (*cf.*, Figure 4.20).

According to Ref. [33], the active area of the CHSPAD has a diameter of about 6 µm. The theoretical peak concentration factor is the relation of the area of single micro-lens to the active area of a single pixel

$$\text{CF}_{\mu\text{L}_{\text{max}}} = \frac{\pi(24\,\mu\text{m}/2)^2}{\pi(6\,\mu\text{m}/2)^2} = 16.$$

From simulations, a factor of about 10 is expected [169]. As CHSPAD *C21* has been hand-selected for best performance among all sensors with microlenses, it might have an above-average PDP. On the contrary, CHSPAD *D1* shows lots of defects, so its PDP might be below average. The higher DCR of the sensor without microlenses (see above) disproportionately reduced the denominator of equation (4.13). The measured signal was only four times higher than the DCR; for the sensor with microlenses, this was about 100-fold. If both chips had had the same PDP, the relatively high concentration factor shown above could be explained with an overall smaller active area of the single SPAD ($d_{\text{SPAD}} \simeq 5\,\mu\text{m}$ for $\text{CF}_{\mu\text{L}} = 22$).

Influence of the microlenses on SPIM-FCS measurements

Microlenses have a significant influence on the number of detected photons due to their ability to concentrate light. In the following the influence of the microlenses on SPIM-FCS measurements is tested.

For this purpose, *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads were used as a sample (for details on the sample see section 2.2.4). First, the fluorescence intensity was recorded by the sensor with microlenses, *C21*, which was then exchanged by the sensor without microlenses, *D1*. The sensors were re-aligned prior to each measurement. Everything else in the SPIM setup was left unchanged between the measurements. The EMCCD camera was used in parallel (50:50 beam splitter) for alignment and control. The laser power at the lightsheet was approximately $I_{\text{LS}} = 80\,\text{W}/\text{cm}^2$.

Figure 4.28 shows the ACCs for two single pixels taken either by the sensor with (blue) or without (red) microlenses. In the latter case, the noise on the ACC was significantly increased. Also, the detected count rate was about 12 times less (0.5 kHz instead of 6 kHz). This drop in the count rate is in good agreement with the expected gain factor of approximately 10 to 20. With the decrease of detected photons, the afterpulsing becomes more prominent. This can be seen as a steep increase of the ACC towards small lag times $\tau < 10\,\text{ms}$.

Figure Figure 4.29 shows the distribution of the measured diffusion coefficients of the sample for both sensors. Data was obtained by a global fit (see chapter 6 for details), a fit of the ACCs only did not yield any meaningful result for the CHSPAD without microlenses. In the case of the sensor without microlenses, the mean diffusion coefficient for all pixels was $D_{\text{no }\mu\text{-l}} = (3.6 \pm 2.4)\ \mu\text{m}^2/\text{s}$. For the sensor with microlenses, a GAUSSIAN fit to the distribution of all pixels' diffusion coefficients resembled $D_{\mu\text{-l}} = (4.1 \pm 0.3)\ \mu\text{m}^2/\text{s}$.

Both mean values of the measured diffusion coefficients are comparable and in agreement with the theoretical value of $(3.85 \pm 0.63)\ \mu\text{m}^2/\text{s}$ [154]. The increased PDP of the CHSPAD with microlenses

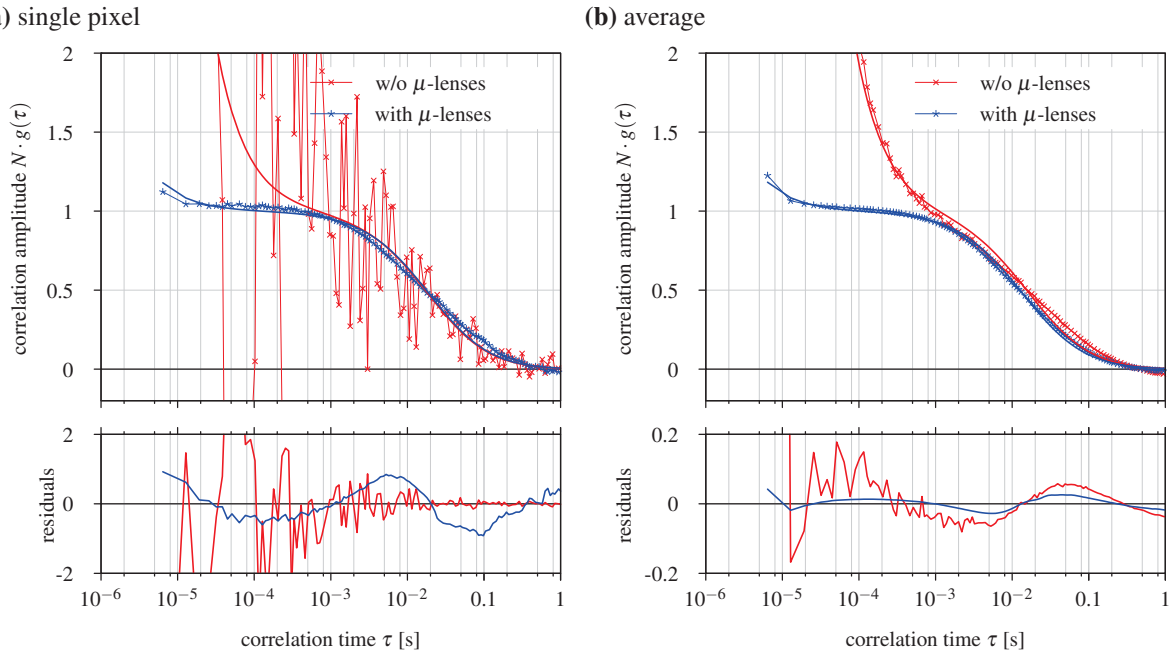


Figure 4.28.: Autocorrelation curves including fits of *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads detected by CHSPAD with and without microlenses. (a) Representative example of a single pixel. (b) Mean value of all pixels. Data obtained from the sensor with microlenses (*C21*) is shown in blue, otherwise (*D1*) shown in red. Dots with thin lines resemble correlation estimates, corresponding fits are shown as solid lines. Data is normalized to $N = 1$ and corrected for an offset.

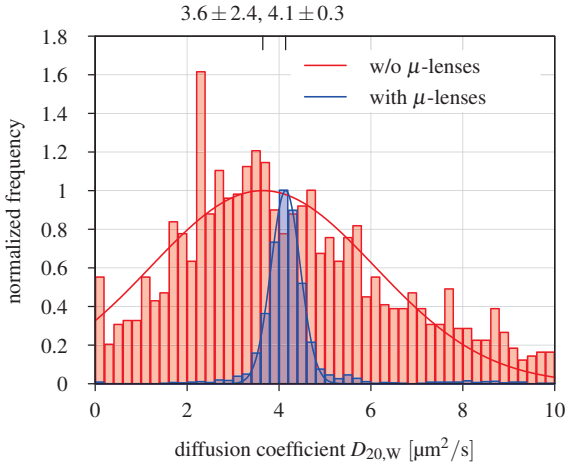


Figure 4.29.: Distribution of diffusion coefficients of a *TetraSpeck* beads sample a measured by a CHSPAD with and without microlenses. Data was obtained using a global fit (see chapter 6 for details). A GAUSSIAN fit of the distribution is shown as a solid curve. Mean values from fit are shown above the plot.

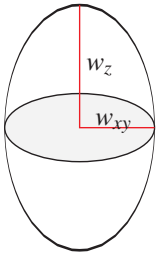


Figure 4.30.: Schematic drawing of the focus GAUSSIAN with an ellipsoid shape in a confocal microscope.

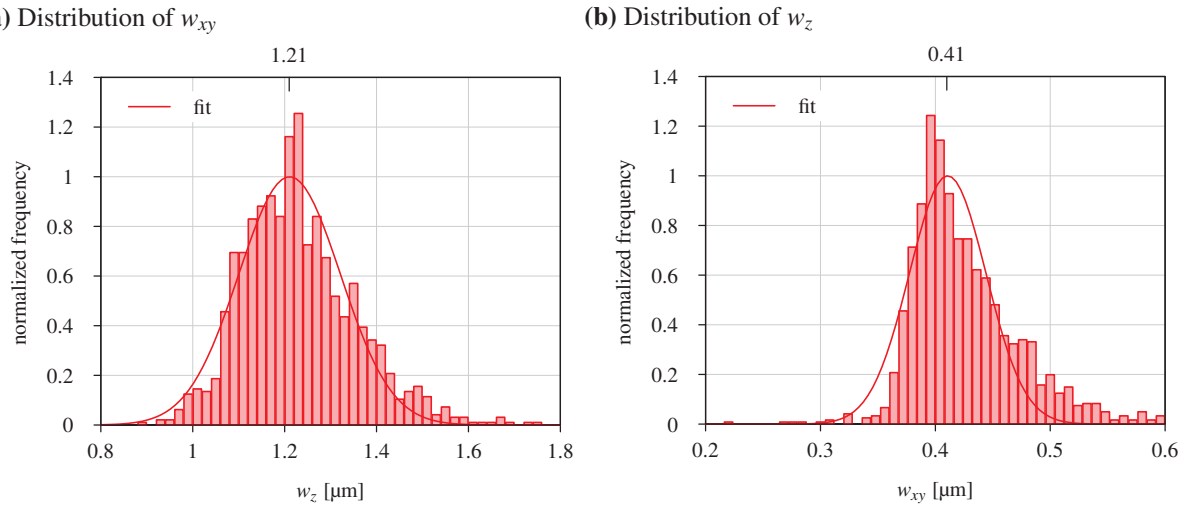


Figure 4.31.: Histogram of the focal volume parameters w_{xy} and w_z as obtained from a bead-scan. Data is normalized to the amplitude obtained from GAUSSIAN fit. Fit results are shown above the plot.

lead to an eight-fold decrease in the width of its distribution. More details on the measurement procedure are presented in chapter 6.

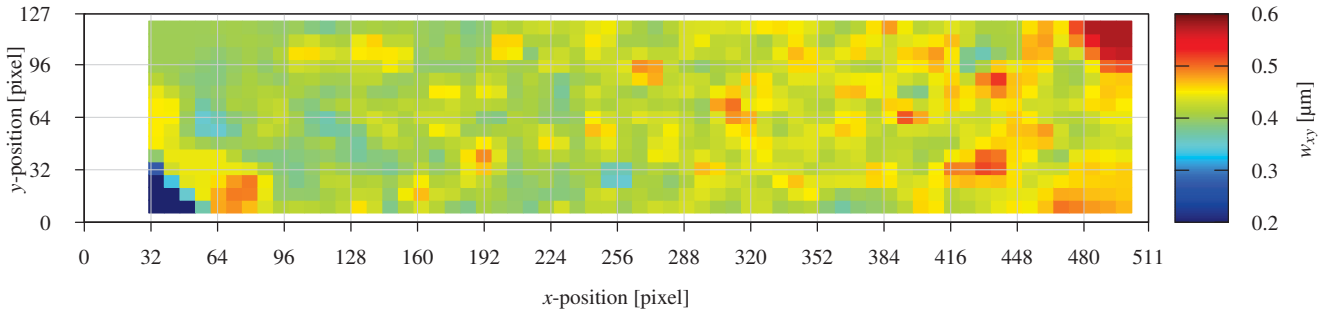
4.3.8. Determination of the size of the focal volume at each pixel

Important parameters for the determination of the diffusion coefficients and the particle concentrations in FCS measurements are the size and shape of the focal volume. In imaging fluorescence correlation spectroscopy (imaging FCS), each single pixel (x,y) in combination with the detection optics defines a virtual focal volume in the image plane. Assuming each focus to have an ellipsoid shape with two equal axes, two parameters are required for its description: the lateral $1/e^2$ half width $w_{xy}(x,y)$ and the longitudinal $1/e^2$ half width $w_z(x,y)$, which are shown in Figure 4.30. Typically, for detectors with fewer pixels, these parameters are assumed to be constant for the entire detection array ([119]). The four-fold larger field of view of the CHSPAD compared to the EMCCD required a more detailed analysis.

An adequate method to determine the focal shape is a bead-scan as described in section 2.3.4. Figure 4.31 shows the resulting histograms of both focal parameters w_{xy} and w_z. Data was obtained from a single bead-scan with 1001 frames and a step width of 200 nm in z-direction. The distribution of w_{xy} were in good agreement with a GAUSSIAN distribution, whereas the distribution of w_z is slightly skewed, which might indicate a position dependence.

The corresponding maps of the parameters across the whole sensor area are shown in Figure 4.32. The distributions of the parameters along the axes are shown in Figure 4.33. The lateral width w_{xy} seemed to be distributed evenly, only with a slight increase in the absolute value along the x-axes (Figures 4.32a and

(a) Map of w_{xy} across CHSPAD sensor



(b) Map of w_z across CHSPAD sensor

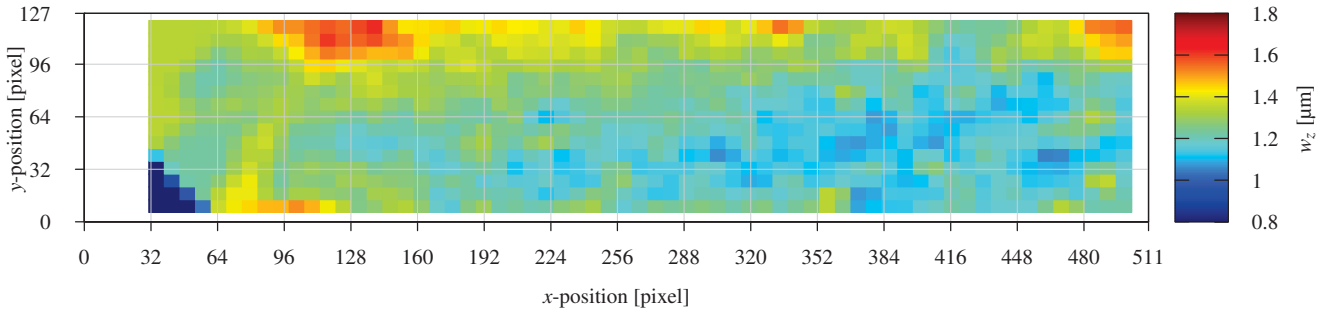


Figure 4.32.: Distribution of focal volume parameters w_{xy} and w_z across the CHSPAD with microlenses obtained from bead-scan. Scattered values from detected and fitted beads were binned into 8×8 tiles. In total, 1402 beads were found in the z-stack. The majority of the beads was detected in the center area.

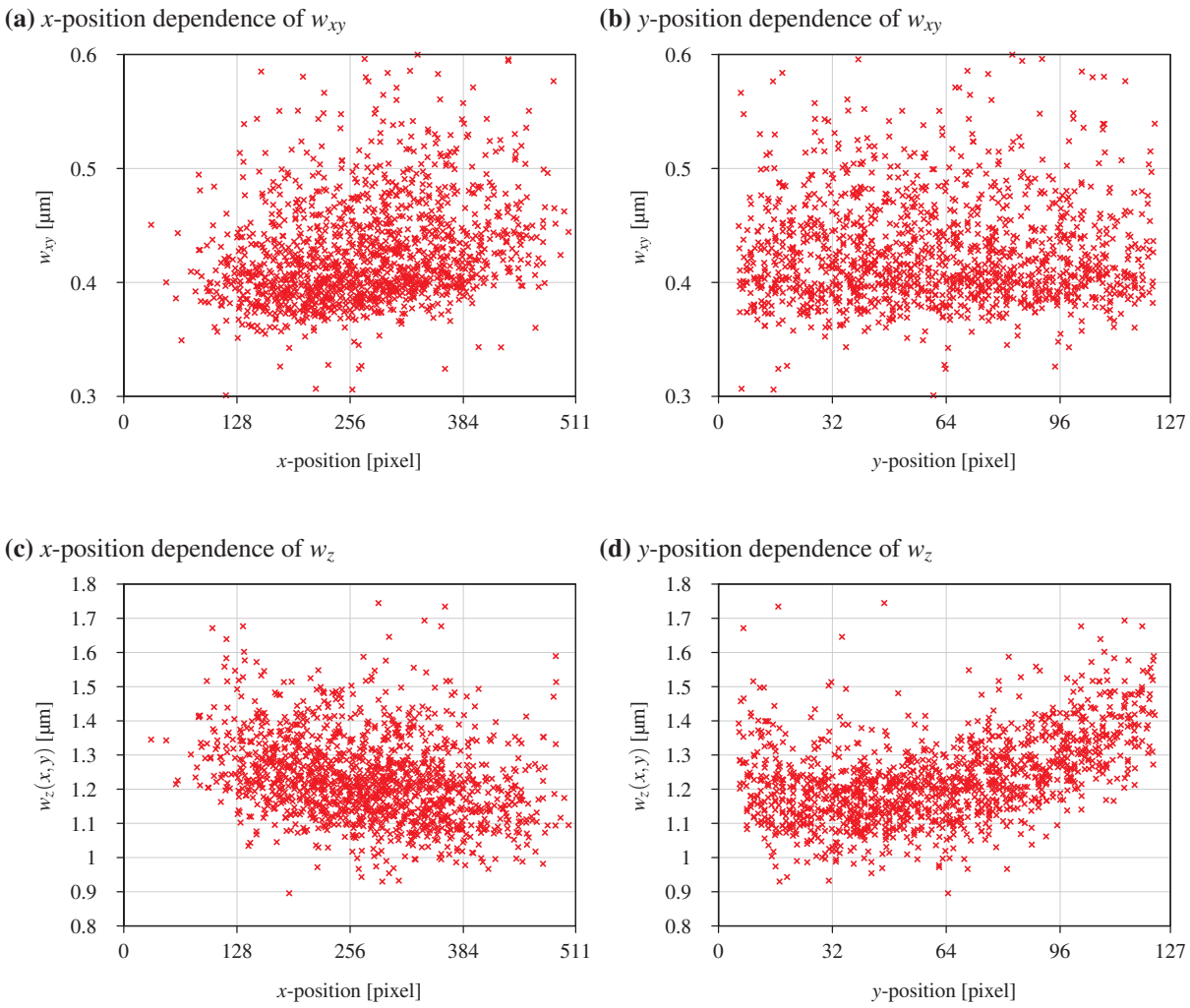


Figure 4.33.: Distribution of the focal volume parameters w_{xy} and w_z along the x and y axis as obtained from bead-scan. In (a) and (b) position dependence on the x and y axis of w_{xy} is shown. The position dependence of w_z is shown in (c) and (d).

4.33a). The second parameter, w_z, seemed to have a minimum in the center of the sensor with an increase towards the edges (Figures 4.32b and 4.33d), which can be explained with the shape of the lightsheet. Additionally, a slight decrease along the x-axes was visible (Figure 4.33c).

It can be stated, that the distribution of the parameters of the focal volume, as obtained by a bead-scan, followed GAUSSIAN distributions. A significant position dependence, except for the shape of the lightsheet was not detected. Further investigations of the homogeneity of the size of the focal volume based on the evaluation of correlation curves of a known sample is shown in appendix E.1.

4.3.9. Temperature of the sensor

In contrast to scientific cameras, the CHSPAD did not allow for any efficient cooling. Figure 4.34 shows the progression of the temperature for a 110 s measurement. To slightly decrease the temperature, a fan was placed below the *Broaddown 4* board to blow air through the gap between both boards. During the measurement, the temperature that was measured on the back of the daughter board rose by about 6 K. Without an ongoing measurement, the sensor had a temperature of about 32 °C, which was approximately 10 K above room temperature. This increase in temperature is mostly due to the photon induced avalanches in the photodiodes of the sensor. Due to the quenching, heat dissipates directly in every single pixel.

The design of the CHSPAD daughter board and its connection to the *Broaddown 4* evaluation board did not allow for placing any heat-conductor in between. Unfortunately, heat dissipating electronics (high-speed clock drivers) were located on the *Broaddown 4* directly below the CHSPAD. In addition to the heat dissipation of the electronics of each pixel, every single avalanche generates heat in the quenching circuitry. Thus, the heat quantity depends on the incident light intensity.

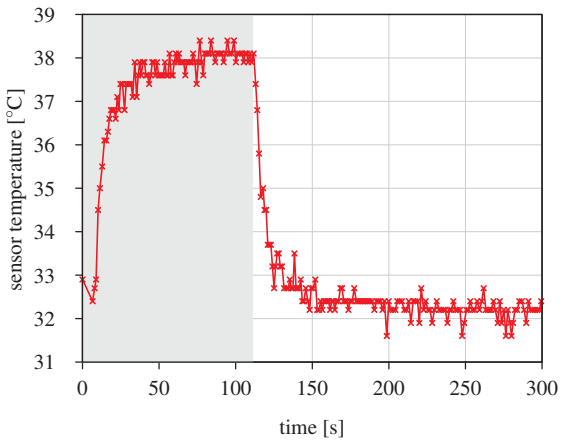


Figure 4.34.: Progression of the temperature of the CHSPAD sensor for a typical measurement. The duration of the measurement is shown as a light gray area. The temperature was measured by a temperature sensor (*Pt100*) attached to backside of the daughter board behind the CHSPAD sensor.

4.4. Conclusion

For both different SPAD arrays, the *RADHARD2* and the CHSPAD, readout hardware was developed, that allows frame acquisition above 100 kHz and camera-like usage of the sensors within the microscope setup. For the CHSPAD, the existing firmware and readout software could be extended to support features that are necessary in the daily routine when used in a microscope. Examples are the support for a region of interest or a hardware-based correction of defective pixels. Moreover, a readout system was designed that allows real-time data acquisition from the CHSPAD at full frame rate. To reduce the overall data rate, lossless compression based on entropy encoding can be efficiently used for the compaction of the raw data. Further improvements of the control signal patterns were done to reduce the significant amount of afterpulsing seen in this sensor.

The CHSPAD array proved advantageous in comparison to commonly used EMCCD cameras regarding the signal to noise ratio in imaging mode, which is expected for this type of sensor as it should behave close to an ideal photodetector. Microlenses, that were attached to every individual pixel of a second CHSPAD chip, led to a more than ten-fold gain in detected light intensity. Several measurements suggested that such microlenses do not have a negative effect on FCS measurements. The only drawback of microlenses were, that if they are not specifically corrected for a certain optical setup, as they are for the SPIM setup, a significant decrease in sensitivity towards the edges of the active area can occur. Except for a position dependence of the depth of the focal volume w_z due to the shape of the lightsheet, no significant position dependency of the parameters of the focal volume were detected.

In summary it can be stated, that both sensors, but especially the CHSPAD with microlenses, are in principle suitable for SPIM-FCS measurements. This is further evaluated in chapter 6 on the basis of commonly used dyes.

5. Implementation of fast signal correlation analysis

5.1. Introduction

In this chapter, the implementation of different multiple-τ correlation algorithms as introduced in section 2.4.2 is described. As hardware platforms for the implementation of the correlators, two recent CPUs, a field programmable gate array (FPGA), and a graphics processing unit (GPU) were evaluated.

5.1.1. Initial situation

When the project was started, a software implementation of a multiple-τ correlator already existed in the evaluation software *QuickFit3* used for the microscope (see section 2.3.3). This correlator is still available (*Multi-Tau 1*, see *QuickFit3* on-line help for details). The overall performance was roughly 1.4 Mflops/s* which was even too slow for real-time correlation using the *RADHARD2* system (at least 102.4 Mflops/s would have been required). A significant amount of time (about 2 %) was spent for initial data conversion from raw binary data to floating-point values. The conversion was done prior to correlation, so on-line calculation of streaming input data was not possible, and a significant amount of memory was required. Furthermore, the existing code does not make use of any optimization, such as single instruction, multiple data (SIMD) instruction set extensions or threads. The tested code was written with versatility in mind and does allow background-correction and bleach-correction. Its main targets were image series, taken with a conventional scientific camera with much lower frame rates and a 12 . . . 16 bit data format.

5.1.2. Requirements of the implementation

A major goal of all implementations in this chapter was the possibility to provide the results of the correlation in real time. This could be used to provide the operator of the microscope with spacial resolved maps of the ongoing diffusion within the sample. That way, he or she can get a brief impression whether the observed sample, *e.g.*, a cell, contains the molecule of interest in an appropriate concentration. Based on this preview, a rough estimation of the sample quality can be done and measurements can be limited to particularly promising samples, which saves evaluation time and disk space.

Real time correlation also gives operator of a microscope the feedback that is needed to align the instrument properly. Singh [200, p. 61] showed, that when the lightsheet and the focal plane of the detection objective overlap perfectly, the resulting diffusion coefficient of the observed species is maximal and the particle number is minimal. So, using the results of a correlation analysis, the alignment can be automated and an overall higher accuracy and better signal quality can be achieved.

Depending on the samples used, typical measurements have a duration in the order of seconds to minutes. With the worst case data rate of about 1.2 GiB/s for the Swiss single photon avalanche diode array (CHSPAD), the amount of raw data would exceed the available RAM of almost any computing platform within seconds. Therefore, the algorithm should support data streaming and should not rely on the entire dataset. Also, the amount of working memory used per correlator should be kept as small as possible, because several thousand correlators can be run in parallel.

*Tested with a 450 MB dataset taken with the *RADHARD2* system comprising 12.6×10^6 frames of 32×8 pixels. Correlation time was about 40 min, when running on an INTEL *i5-2520M* 2.5 GHz (turbo: 3.2 GHz).

5.1.3. Properties of the correlation function

An important feature of the autocorrelation function is that processing of several pixels can be trivially parallelized, because pixel do not share data. This is not valid if a spatial cross-correlation function is calculated. But then, the FPGA used for readout can also be used to presort data, split and duplicate the data (*e.g.*, for cross-correlation), so that the correlators do not have to pre-process their input.

The multiple-τ correlation algorithm used in this thesis allows for a further parallelization on the level of linear blocks, resulting in a series connection of linear correlators. Input data is then flowing between such blocks, while passing through the entire correlator.

If only a few pixels are measured, *i.e.*, with photomultiplier tubes (PMTs) or single photon avalanche diodes (SPADs), typical frame rates range from 1×10^5 to 1×10^9 fps. For cameras with thousands of pixels, the frame rate usually drops down to 1×10^3 fps. Implementations proposed so far (see section 3.2 for an overview) typically focused on efficient correlators with only a few channels (*i.e.*, pixels). With new SPAD arrays, which comprise several thousands of pixels, a completely new implementation was required.

Another important property is the format of the input data, that is, whether the detecting device outputs simple photon / no-photon data (black and white, *i.e.*, SPAD arrays) or accumulated values (gray-scale, *i.e.*, CMOS cameras). If only binary data with single bit information (*e.g.*, no photon / ≥ 1 photon) is processed, data paths can be optimized accordingly. This is especially relevant for hardware-based implmentations. Correlators based on CPUs or GPUs typically use a generic data type with a constant width of the data path (*i.e.*, 32 bit-wide single precision floating-point values).

5.2. Mapping the multiple-τ correlator algorithm onto different hardware platforms

For all multiple-τ algorithms in this thesis, in between two linear correlator blocks, two successive values have been averaged (*i.e.*, $m = 2$ in section 2.4.2). The calculation of the correlation function is typically split into two steps. First, $g_{\tau_k} = \langle I(t) \cdot I(t+\tau_k) \rangle$ (the nominator in equation (2.30)) is evaluated and finally the full correlation curve is calculated by normalizing g_{τ_k}. The monitor channel $M = \sum_t I(t)$ (*cf.*, equation (2.30)) is calculated in parallel in the first stage of the correlator, the so called 'front end'.

The most recurring operation in the calculation of the autocorrelation function (equation (2.30)) for a specific delay time τ_k (*i.e.*, a single lag) is to multiply two intensity values I and accumulate the results, a multiply accumulate (MAC) operation:

$$(G_k)_n = I_n \cdot I_{n-k} + (G_k)_{n-1} \,. \tag{5.1}$$

This operation has to be repeated for all n input values in every single lag of each block. It perfectly maps to dedicated functional units found in recent high performance hardware, such as arithmetic logic units (ALUs) of modern CPUs, MAC units on signal processors, or DSP slices found in FPGAs. If the underlying data type is a floating-point value, this is equal to two floating-point operations (FLOPs) or a single FMA-FLOP, since some modern CPUs can execute this as a single operation (see [174] for a detailed description). In case of the FPGA with no native support for floating-point operations, MAC operations are equated with FMA-FLOPs for comparability.

Depending on the target architecture, two main optimizations can be done, determined by the amount of available memory: If only a small amount of RAM is at hand, the output data stream of a single block cannot be saved and the consecutive blocks have to be executed in an increasing order. Input data is then propagated through the entire correlator until it is consumed (*i.e.*, by an accumulation step between two blocks) and the next input value can be processed. This way, all lags of all correlator blocks must be present in memory at all times.

input: input values of the first channel $I^u_0 \dots I^u_{N-1}$,
input values of the second channel $I^d_0 \dots I^d_{N-1}$
data : accumulators for each lag $\mathsf{ACC}[N_{\text{blocks}}, N_{\text{lags}}]$,
delay registers for each lag $\mathsf{REG}[N_{\text{blocks}}, N_{\text{lags}}]$,
storage for output values of each block $I^u_{\text{old}}[N_{\text{blocks}}]$ and $I^d_{\text{old}}[N_{\text{blocks}}]$,
the current intensity values that are processed I^u and I^d

```
1  for n ← 0 to N − 1 do                                   /* iterate over all input values */
2      I^u ← I^u_n;
3      I^d ← I^d_n;
4      for b ← 0 to N_blocks − 1 do
5          for l ← 0 to N_lags − 1 do                      /* process all lags in current block */
6              ACC[b,l] ← I^u · I^d + ACC[b,l];
7              SWAP (REG[b,l], I^d);                       /* store current and load old value */
8          end
9          if block was executed an even number of times then   /* run next block */
10             I^u ← I^u + I^u_old;
11             I^d ← I^d + I^d_old;
12         else                                            /* store values for next iteration and stop processing */
13             I^u_old[b] ← I^u;
14             I^d_old[b] ← I^d;
15             stop processing of b, continue with next n;
16         end
17     end
18 end
```

Algorithm 5.1: Naïve algorithm of a multiple-τ correlator. The algorithm is used to calculate the correlation function estimates for a multiple-τ correlator with B blocks ($b = 0 \dots N_{\text{blocks}} - 1$) comprising L lags each ($l = 0 \dots N_{\text{lags}} - 1$). The correlator accepts two input values I^u_n and I^d_n in each cycle n. If used as an autocorrelator, set $I^u_n = I^d_n$. In each lag an accumulator *ACC* holds the result of the specific correlation. Values in the delayed data path are stored per lag in *REG*. The actual correlation is performed in line 6 on the current values I^u and I^d. If a block has been processed an odd number of times, the resulting output values are stored for each block in I^u_{old} and I^d_{old}. In case of an even number of executions, the next block is processed using the sum of the current values I^u and I^d and the stored values I^u_{old} and I^d_{old} as inputs. This algorithmic representation is optimized for a low usage of registers. After all N input values have been processed, the correlator needs to be flushed, *i.e.*, executed with zeros as input values, until all delayed values have passed through all lags.

If a large amount of memory is available, a certain linear block of the correlator can be processed for multiple input values. Its output is buffered in memory and is later consumed by the next correlator block. In the latter case, only a small amount of local memory (*e.g.*, register file or block RAM (BRAM)) is needed to hold the accumulators of a single block as well as the values in the delayed data path.

5.2.1. Naïve implementation of the correlation algorithm

The initial implementation that was done for the CPU correlator is based on the 'naïve algorithm', which is shown in Algorithm 5.1. It is optimized for a small memory footprint.

The basic idea is to propagate a single input value through all lags until it is consumed by an accumulation step between two correlator blocks. The algorithm is optimized for low register usage so that ideally all accumulators of a single block fit into the register file (commonly the 16 MMX registers of a CPU with SIMD instruction set extension, *cf.*, section 2.6.1). With the help of 'vertical unrolling' (*i.e.*, the pro-

input: input values of the first channel $I^u_0 \dots I^u_{N-1}$,
input values of the second channel $I^d_0 \dots I^d_{N-1}$
data : accumulators for each lag $\mathsf{ACC}[N_{\text{blocks}}, N_{\text{lags}}]$,
delay registers for each lag $\mathsf{REG}[N_{\text{blocks}}, N_{\text{lags}}]$,
storage for output values of each block $I^u_{\text{old}}[N_{\text{blocks}}]$ and $I^d_{\text{old}}[N_{\text{blocks}}]$
the current intensity values that are processed I^u and I^d

```
1  for n ← 0 to 2·N − 1 do                          /* iterate over all input values */
2      b ← getBlock(n);                              /* determine current block */
3      if b = 0 then                                 /* load single pair of input values */
4          I^u ← I^u_{n/2};                          /* either directly from input... */
5          I^d ← I^d_{n/2};
6      else
7          I^u ← I^u_old[b − 1];                     /* ...or from value store */
8          I^d ← I^d_old[b − 1];
9          I^u_old[b − 1] ← 0;                       /* reset, so a simple addition can be used */
10         I^d_old[b − 1] ← 0;
11     end
12     for l ← 0 to N_lags − 1 do                    /* process all lags in current block */
13         ACC[b,l] ← I^u · I^d + ACC[b,l];
14         SWAP(REG[b,l], I^d);                      /* store current and load old value */
15     end
16     I^u_old[b] ← I^u_old[b] + I^u;                /* store values for following block */
17     I^d_old[b] ← I^d_old[b] + I^d;
18 end
```

Algorithm 5.2: FPGA-optimized version of the naïve algorithm for a multiple-τ correlator. In contrast to the naive implementation (*cf.*, algorithm 5.1), the order of blocks is determined by the block scheduling algorithm (line 2) described in section 5.2.4. This requires to run two linear blocks of the correlator per single input value. Typically, the entire context is stored in a BRAM of an FPGA. In case of a hardware implementation, the conditional (line 3) can be unified trivially.

cessing of multiple time-steps in parallel, see section 5.2.8) the entire register file can be used efficiently. With slight modifications, this algorithm was also be used for the FPGA-based implementation.

5.2.2. FPGA-optimized implementation

The *LASP* board (see appendix B.1) which was used for the *RADHARD2* and the implementation of an FPGA-based correlator (*cf.*, section 5.5.2) has relatively small memory resources (432 KiB BRAM and 20 MiB RAM for each FPGA). Thus, the algorithm described above was adapted to fit this platform, too.

An advantage of an FPGA, if compared to a CPU, is the large amount of BRAM inside the device, which can be used in the same way as the (smaller) register file of a CPU. This way, data required by a correlator for a single pixel, the so called context, is available without latency.

The naïve algorithm is based on branching, which does not map well to an FPGA. This was overcome in algorithm 5.2 by using a scheduler (line 2) instead of a conditional branch to determine the order of execution of the linear blocks within the correlator. By running two linear blocks per single input value, it is possible to perform the processing of all blocks in the correct order (see section 5.2.4 for details on the scheduling). In contrast to the naïve algorithm, this solution requires for an additional storing of the output values in each iteration.

input: buffer for input values $I_v^u[0,0\ldots N_c-1]$, $I_v^d[0,0\ldots N_c-1]$
data : accumulators for each lag $\mathrm{ACC}\left[N_{\mathrm{blocks}},N_{\mathrm{lags}}\right]$,
delay registers for each lag $\mathrm{REG}\left[N_{\mathrm{blocks}},N_{\mathrm{lags}}\right]$,
output buffers for intensity values $I_v^u[N_{\mathrm{blocks}},N_{\mathrm{chunks}}]$ and $I_v^d[N_{\mathrm{blocks}},N_{\mathrm{chunks}}]$,
counter for invocation of linear blocks $\mathsf{ticks}[N_{\mathsf{blocks}}]$,
the current intensity values that are processed I^u and I^d

```
1  while input data available do
2      b ← getBlock();                          /* determine next block to process */
3      for c ← 0 to N_c − 1 do                  /* process a full block of input values */
4          I^u ← I_v^u[c,b];                    /* load input data...  */
5          I^d ← I_v^d[c,b];
6          I_v^u[c,b] ← 0;                      /* ...and reset */
7          I_v^d[c,b] ← 0;
8          for l ← 0 to N_lags − 1 do           /* process all lags in current block */
9              ACC[b,l] ← I^u · I^d + ACC[b,l];
10             SWAP(REG[b,l], I^d);             /* store current and load old value */
11         end
12         p ← c/2 + MOD(ticks[b+1],2) · N_c;   /* calculate output data position */
13         I_v^u[b+1,p] ← I_v^u[b+1,p] + I^u;
14         I_v^d[b+1,p] ← I_v^d[b+1,p] + I^d;
15     end
16     ticks[b] ← ticks[b] + 1;
17 end
```

Algorithm 5.3: Streaming algorithm for a multiple-τ correlator. Input data must be provided in multiple buffers $I_v^{u,d}[0,0\ldots N_c-1]$, and might be written asynchronously in parallel, if this does not affect the ongoing correlation. In each linear block N_{chunks} input values are processed. The next block to process, b, can be determined by the scheme described in section 5.2.4. The *ticks* counters are used to determine the output locations for the data, and whether the first or second half is written. After being read the input data must be zeroed as it is used to add up the output values (line 13). Vertical unrolling and input data conversion are not shown.

5.2.3. Streaming correlator algorithm

When implementing the multiple-τ algorithm on a GPU, the significantly different structure required for a different correlation algorithm: A GPU is typically characterized by a large amount of memory that is connected to the processing units with a high latency (several hundred clock cycles, no cache). The multiprocessors of a GPU have a rather large register file (in comparison to a CPU) to balance the memory interface characteristics. Over all, the GPU architecture is optimized for streaming large quantities of data, while applying highly parallelized transformations to it. Such problems are often encountered in three-dimensional computer graphics. For the implementation of a multiple-τ correlator, this implies that the number of context switches, which break the dataflow, had to be minimized.

The solution shown in Algorithm 5.3 is optimized for a large amount of memory with a high latency and a small register file, thus fitting the needs of the GPU. A schematic drawing of the dataflow is shown in Figure 5.1. As for the FPGA-optimized implementation, the order in which correlator blocks are processed is determined by the scheduler (line 2). Again, alternately with a higher order block, the first order block is executed.

If a large amount of memory is available, which applies particularly for GPUs, a single block of the correlator can be processed for a long time. The input and output is buffered in memory and switches to

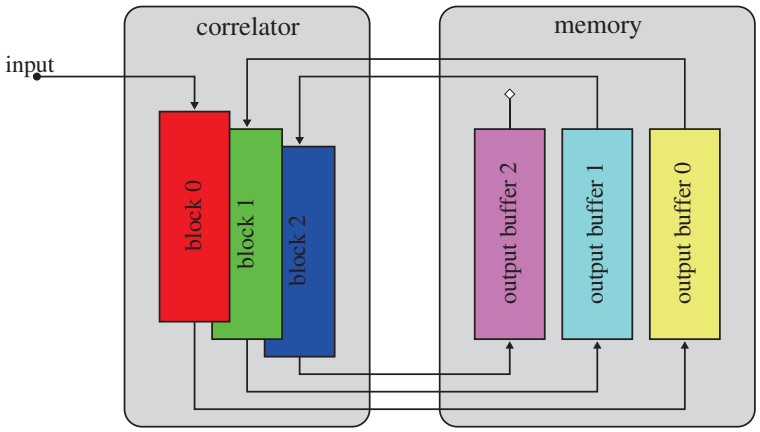


Figure 5.1.: Schematic drawing of the streaming correlator algorithm. Only one single linear block of the correlator is executed at every instant. Input and output is read from and written to buffers residing in the memory. The order of the blocks is determined by a scheduler.

another correlator block are reduced. The relatively small amount of data in the context (typically eight accumulators and the same number of delay values) can be held in the register file all the time. If free registers remain, these can be used for further optimizations, such as the combination of two correlator blocks into a single one (this was also done for the GPU correlator; see section 5.4.2).

During the design of the GPU correlator, it was found that this streaming algorithm was also quite efficient on the CPU (see section 5.3.4 for details on the implementation). Also the proposed implementation of the FPGA-based correlator following the dataflow paradigm is based on this algorithm (section 5.5.7).

5.2.4. Block scheduling algorithm

As shown in equation (2.23), the execution time of a multiple-τ-correlator ($m = 2$) is just twice the execution time of the first linear correlator block times the number of its invocations. So when the operation of the entire correlator is divided into time slots, every second slot is devoted to the first block, every fourth to the second block and so forth. For both, the FPGA and for the GPU implementation of the correlation algorithm, an efficient scheme had to be found that determines the order of blocks and only depends on a counter c of the already processed blocks. Further, this scheduling scheme had to guarantee that a linear correlator block b is only executed after its predecessor $b-1$ has been executed twice. In Algorithm 5.2 and Algorithm 5.3 this scheduling was denoted as a call to `getBlock()`. Its implementation is based on the following relations between the block number b and the counter c, mentioned above:

$$\begin{aligned} b = 0 &: \quad c \bmod 2^1 = 0 \\ b = 1 &: \quad c \bmod 2^2 = 3 \\ b \geq 2 &: \quad c \bmod 2^{b+1} = (2^b - 3) \end{aligned} \tag{5.2}$$

Figure 5.2 shows the order of blocks b that are processed in 32 time steps ($c = 0 \ldots 31$). If c is represented as a binary number, certain patterns in its digits become apparent, which are summarized in Table 5.1 for the first eight blocks. For example, the linear correlator $b = 0$ is executed in every second cycle when the last digit of c is '0'; the block $b = 1$ is executed whenever c ends with '11' and so forth. This scheme can be used to directly implement the scheduler in hardware or software. In hardware (*i.e.*, on an FPGA) this is especially efficient, since its implementation is based on comparison operations only.

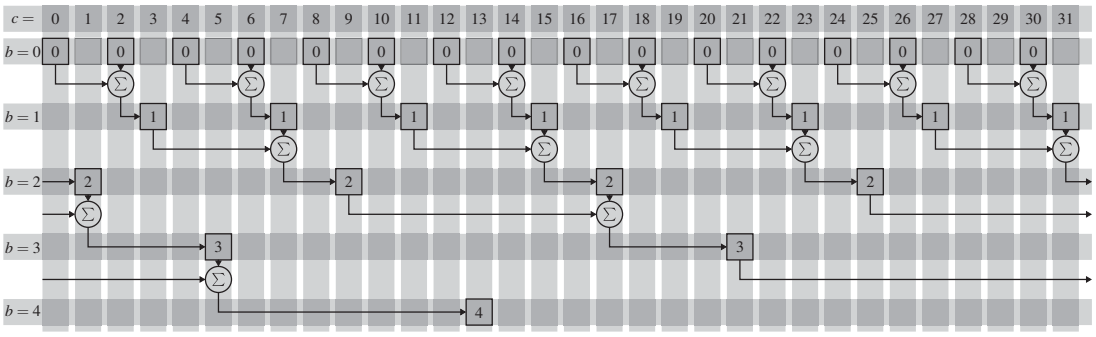


Figure 5.2.: Illustration for the block scheduling algorithm for the multiple-τ correlator. The counter c in the top row counts the number of already processed linear blocks. Below that the order of processing is illustrated: When a block has been processed twice, the following block, $b+1$, can be processed using the summed output of its predecessor. $\sum$: output summation.

block b	binary representation of the counter c								
	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	⋆	⋆	⋆	⋆	⋆	⋆	⋆	⋆	0
1	⋆	⋆	⋆	⋆	⋆	⋆	⋆	1	1
2	⋆	⋆	⋆	⋆	⋆	⋆	0	0	1
3	⋆	⋆	⋆	⋆	⋆	0	1	0	1
4	⋆	⋆	⋆	⋆	0	1	1	0	1
5	⋆	⋆	⋆	0	1	1	1	0	1
6	⋆	⋆	0	1	1	1	1	0	1
7	⋆	0	1	1	1	1	1	0	1

Table 5.1.: Block scheduling algorithm. Binary representation of counter c that solves equation (5.2) for a given linear correlator block b. ⋆ denotes a don't care condition.

5.2.5. Data types

The basic data type used for the all correlators is the 32 bit wide IEEE 754-2008 [2] single precision floating-point number. Solely the FPGA implementation is based on 32 bit integer values for the accumulator and a 16 bit-wide data path.

5.2.6. Performance considerations

To handle 156 250 frames per second, the frame rate of the CHSPAD with 512×128 pixels, 10.2×10^9 pixel have to be processed by the correlator per second. When a new input value arrives for a pixel, it has to be shifted (*cf.*, the illustration of the multiple-τ correlator as a shift register in Figure 2.27) into the first linear correlator block, which then has to execute 8 MAC operations, one for each lag. To process all further blocks, another 8 MAC operations have to be executed. This surprising finding of only 16 MAC operations used per input value for processing all blocks is explained by the averaging between linear correlator block, and is detailed in section 2.4.2. Therefore, in order to process all pixels of the CHSPAD sensor at the maximum frame rate of 156 250 fps in real time, a total performance of 164 Gflops/s (FMA) is required.

5.2.7. Memory bandwidth and performance requirements

For a single precision floating-point implementation of the correlator, the first block has to expand single bit input values to the 32 bit representation of floating-point values used for block processing. In case of a cross-correlator, the input data has to be expanded to even to 64 bit, as two different input values have

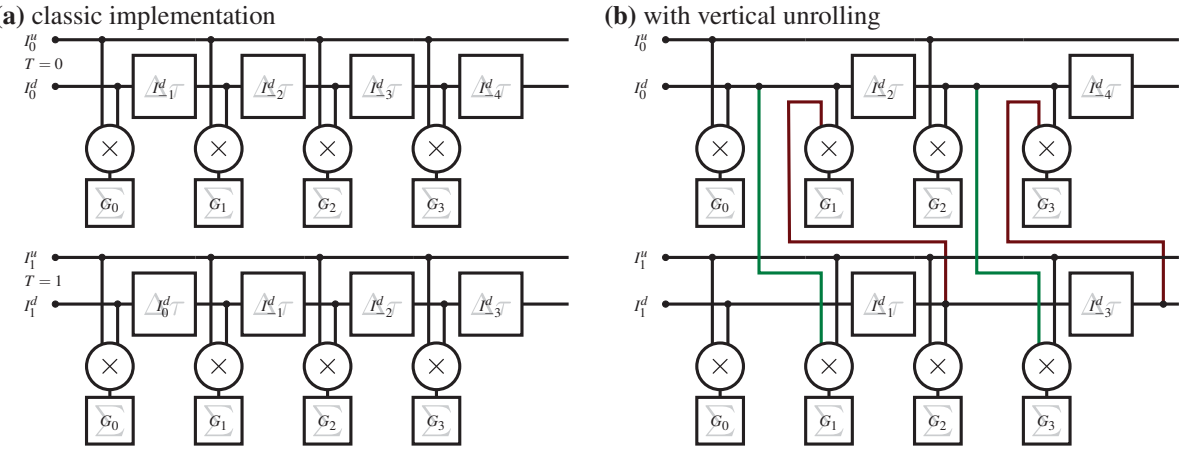


Figure 5.3.: Concept of vertical unrolling of a linear correlator. In (a) a 'classic' linear correlator is shown for two time-steps. By executing two time-steps in parallel, a vertical unrolling of factor 2 (b), the number of delay elements can be halved. See Figure 2.27 for the schematic of a multiple-τ correaltor.

to be processed. For a typical raw dataset of 4 GiB, this will augment the dataset to at least 128 GiB for single precision floating-point values to be processed and would typically not fit into a current computer's or GPU's memory. In case of double precision floating-point values, the amount is again doubled. Thus, the processing of the data requires a slicing of the raw data into chunks that can fit the available RAM.

If the data type across all blocks is the same, each block of a multiple-τ correlator reduces its input dataset by a factor of two (due to the averaging step) before handing it over to the next block. According to the scheduling scheme described above, the first order correlator block is run once per input frame, reading the raw data and storing expanded data for the next block. Additionally, a second higher order block is processed. Thus, for each frame the streaming algorithm requires about 1 MiB of combined data transfers from and to memory*.

For a correlator with an adaptive data path, where each block has only the required bus width this effect is less significant: Although the output data rate is halved by each block, the bus width increases by one bit for each successive block (*i.e.*, first level: one bit, second level: two bit, third level: three bit, and so forth). Thus, for low order blocks the data rate is almost constant and only halves for higher order blocks, where the increase in bus width is less significant.

Therefore, if 156 250 frames of 512×128 pixels shall be processed every second, the streaming algorithm requires an accumulated memory bandwidth of 154 GiB/s. Without taking further optimizations into account (*i.e.*, chaches), only the GPU hardware is suited to achieve this requirement.

5.2.8. Vertical unrolling

The classical approach of a multiple-τ correlator is a series of identical designed lags organized in blocks with some additional summation in between. Without further optimization, five CPU-cycles are necessary for each single lag (*cf.*, algorithm 5.1): load value for summation (s), load stored current value $c \to v_{l,i+1}$, store local value from predecessor ($v_{l,i} \to c$), correlate values, and store s. Modern CPUs can use their load/store engine and floating-point unit (FPU) in parallel, but then four cycles are still necessary.

The basic concept, as shown in Figure 5.3, is to combine a certain number N_{VU} of consecutive time-steps t into one repetition of the inner block. Operations are then performed on the same accumulator s,

* $\underbrace{8\,\mathrm{KiB}}_{\text{1st level, read}} + \underbrace{256\,\mathrm{KiB}}_{\text{1st level, write}} + \underbrace{512\,\mathrm{KiB}}_{n\text{-th level, read}} + \underbrace{256\,\mathrm{KiB}}_{n\text{-th level, write}} \approx 1\,\mathrm{MiB}$

and corresponding loads/stores can be omitted. That way the current value has to be loaded and stored once per lag (see Listing 3).

The drawback of this method is the higher usage of registers, which are typically limited to 16 on current CPUs (*cf.*, section 2.6.1). Despite the sum s, both the local values v_l and the global values v_g have to be held in memory. As N_{VU} also represents the number of MAC operations that can be performed in a row, increasing N_{VU} can significantly improve the performance.

Vertical unrolling is typically limited by the number of registers that are available. If during operation more values are accessed than fit into the register file, some values have to be exchanged with the memory, or at least with the cache.

5.3. CPU correlator

In the following the implementation of the CPU correlator is described. Both algorithms, the naïve and the streaming version, are compared and evaluated on two different CPUs.

5.3.1. CPU platforms

The software implementation was evaluated on an AMD *FX-8320* (*Piledriver* microarchitecture) and an INTEL *i7-4770* (*Haswell* microarchitecture, see appendix A.2 for details on the computer-platforms). Both CPUs feature SSE and AVX instructions for vectorization, as well as fused multiply accumulate (FMA) operations. Further details on the CPUs are given in section 2.6.

Although the AMD *Piledriver* platform features eight cores with dedicated 128 bit soft streaming extension (SSE) units each, 256 bit wide advanced vector extensions (AVX) operations are performed by combining two of those, in a so called module. In contrast to that, the INTEL platform offers twice the performance per GHz as every core comprises two dedicated 256 bit FPUs, so one for each thread (hyper-threading).

5.3.2. Maximum performance considerations

As already pointed out in section 5.2.6, a memory bandwidth of 154 GiB/s is required to achieve real-time performance with the streaming algorithm for the CHSPAD. From this point of view, the implementation should achieve a maximum of 17 % (INTEL) or 20 % (AMD) of the real-time performance (see Table 2.7 for details on the CPUs). Typically, internal caches significantly reduce the number of accesses to external memory, so a higher performance was expected. If the theoretical single precision floating-point performance is taken into account, speed-ups of 1.3× (INTEL) and 0.7× (AMD) over real time should be achievable.

5.3.3. Precision considerations

The CPU correlators are based on the single precision floating-point data type. If a bleach-correction is applied, at least double precision floating-point must be used to retain enough precision.

Results slightly differed between the FMA version of the code and the version without. This divergence originates in the rounding step that is only performed once after the full multiplication and addition in case of the FMA instruction. Classically, rounding is done after each floating-point operation.

Although the correlation operation (*i.e.*, MAC operation) is done on binary input data only, floating-point values were used for the implmentations, due to the better support of this data type in the AVX and SSE instruction set extensions. The single precision floating-point data type with a 23 bit mantissa ($M = 23$) can represent an integer with an absolute value smaller than 2^{24}. This is sufficient for the data path for the typically used 16 linear blocks in a multiple-τ correlator. In case of the accumulators, further

considerations are needed: Due to the summation step after each linear correlator block, which is used for averaging in the multiple-τ algorithm, the width of the data path increases by a a single bit for every subsequent block (*e.g.* 1 bit $\rightarrow$ 2 bit from first to second block and so forth). In turn, the execution rate is halved. For the worst case scenario with a constant input signal equal to '1', the content of the accumulator is increased by $2^{2b}/2^b = 2^{b*}$ in each lag for a linear block $b = 0 \ldots B-1$. This in turn leads to the requirement of an additional bit in the width of the accumulators for each block, to maintain comparable accuracy.

To not overflow the accumulators in a $B = 16$ block multiple-τ correlator, the number of input sample in the last block ($b = 15$) must be smaller than $N < 2^{(M+1)-(B-1)} = 2^9 = 512$ for the worst case scenario described above. As in SPIM-FCS light conditions are typically very dim and also the correlation curves are expected to decay for higher blocks, the results tended to be exact for a significantly higher number of input values. For a random dataset (white noise, 720 000 frames of input data) of which the normalized correlation resembles constant 0, a comparison of the single precision and double precision implementations gave a maximum relative deviation of $\delta_{\text{single, double}} < 4 \cdot 10^{-6}$. The maximum relative deviation of the single precision implementation with FMA and without FMA instructions of the same dataset was $\delta_{\text{single, FMA}} < 4 \cdot 10^{-7}$.

5.3.4. Implementation

For performance evaluations, the naïve algorithm as well as the streaming algorithm were implemented. Both implementations are based on a platform-independent zero-overhead vectorization library (see section 2.6.4) to make use of the most recent SIMD instruction set extensions (*i.e.*, AVX and SSE). These instructions operate on vectors with 4 (SSE) or 8 (AVX) entries instead of scalar values.

Although recent compilers are able to automatically vectorize code within limitations, manual vectorization is usually required to gain the highest possible performance. Depending on the vectorization instruction set, a group of 4 or 8 pixels is processed in a single thread in parallel. For an entire image of the CHSPAD detector (65 536 pixels) several thousand threads are instantiated using the *OpenMP* compiler extension [50]. *OpenMP* then schedules subsets of these threads for execution, so only as many threads as CPU cores run in parallel.

Both algorithms, the naïve algorithm and the streaming algorithm were implemented as described in section 5.2.1 and section 5.2.3. In case of the streaming algorithm, the implementation was split into two distinct routines: one for the first linear block and another one for all other blocks. These two routines are scheduled intermittently as described before. The first routine, the 'front end', also converts binary input data into vectors SSE or AVX vectors.

In order to optimized the MMX-register usage, 'vertical unrolling' was used for both implementations. The correlator code can be configured by several preprocessor macros (*e.g.*, for the level of vertical unrolling, the chunk size, the data reorganization, *etc.*). This reduces the amount of performance limiting branches in the code, but also requires rebuilding the correlator upon changes of the parameters. As the correlator software was also used for data evaluation, additional statistics and calculations of the raw dataset can be generated on demand, including bleach correction.

Low-level optimizations

Although several optimizations were done by the compiler itself, further optimization was still possible. It appeared that manual loop unrolling improves performance significantly, although loop unrolling was explicitly forced by a compile keyword[†]. By explicitly using pointer arithmetic, a further gain in

*For a linear correlator block $b = 0 \ldots B-1$, the maximum value in the data path of a linear block is 2^b, thus the maximum value of the multiplication is 2^{2b}. At the same time the execution rate is reduced by a factor of 2^b.

[†]`(__attribute__((optimize("unroll-loops"))))`

performance was reached.

A crucial precondition for optimal cache performance was the alignment of the vector data in memory. To optimize the CPU's memory access performance, vectors were aligned to 32 Byte boundaries. Several variables were explicitly pinned to the CPU registers, using the type *C* type-modifier `register`. In addition, data locality (and thus memory and cache performance) was optimized by keeping often-used state variables (*e.g.*, the accumulators and the delay registers) in a contiguous memory buffer in alternating order. This way, data can be prefetched by the CPU to the caches in background, and no additional time is spend waiting for memory access.

Input data reordering

The input data usually arrives frame after frame, in row-major order, so the time series of each pixel is not contiguous. For the correlation algorithms, this order is not advantageous, especially for the streaming variant, as this processes data from buffers containing time series of single pixels (or few in a vector). Therefore, a reordering step was introduced, which was done in software, but could potentially also be off-loaded to the FPGA used for data acquisition. If only the runtime of the correlator is considered, a performance gain of roughly 180 % over the implementation without reordering was achieved, as caches were used much more efficiently (see Figure 5.5 for the runtime of the reordering). In the following, time required for reorganization is not taken into account.

Conversion of raw input data

Data conversion of the 8×1 bit input to an AVX floating-point vector of 8×4 Byte can either be done with a lookup table (LUT), where the input data is used as an index, or with a set of integer and logic operations (see appendix D). The latter option proved to be faster, especially for an expansion on an 8 bit input word with a relatively large corresponding LUT (8 KiB), which interferes with the caching efficiency of the correlator.

Flushing the correlator

After all raw input data has been processed by the correlator, a sufficient amount of zeros has to be fed into it so that all input values in the delayed data path propagate through all lags. Without this 'flushing' of the correlator, the full statistical power of the input dataset would not be used for the correlation function estimation. The basic strategy for flushing is to run the entire correlator until a full cycle is reached (*i.e.*, the last block was executed once, the 2nd last block twice, the 3rd last block four times and so forth). Then, the correlator is run once again for an additional full cycle, so that the accumulators in between two blocks release all intermediate results.

The required number of input values (either single frames or chunks, depending on the implementation) that have to be processed to flush the correlator is:

$$n_{\text{input values}} = 2^{B+1} - \left(n_{\text{values processed}} \bmod 2^{B}\right). \tag{5.3}$$

Depending on the number of input values that have to be processed per block, flushing all blocks can take a huge amount of additional processing time, which even might exceed the runtime of the actual correlation. For the GPU correlator that uses larger chunks of input data, a more advanced flushing algorithm was used (*cf.*, section 5.4.2) which one could also use for the CPU implementation.

5.3.5. Performance evaluations

Test conditions

On a CPU that exclusively runs the corrector, the execution time should be the same, whenever the correlator is run. But since the software runs on a system that also executes an operating system (OS) which runs various background tasks that are not related to the correlator itself, a lot of non-determinism will be encountered in such a real-world setting. Additional non-deterministic influences are the interactions between the CPU cores (*e.g.*, cache coherence), the different caches, pipelining of the CPU, etc. All these influences will inevitably lead to a certain jitter on the results of any performance evaluation. An in-depth discussion of these influences can be found in Ref. [231].

For each dataset in the following tests, at least five single measurements were taken into account. Tasks were executed in a way that identical configurations of the correlator were not run successively to account for non-deterministic influences on a rather long timescale (*i.e.*, IO background-tasks). The execution time was measured using the `clock_gettime(CLOCK_MONOTONIC, ...)`, which in principle provides nanosecond resolution. Compared to the `gettimeofday()` function, it tends to be more accurate. If not stated differently, measurements were done using $N_{\text{threads}} = 8$ threads and a chunk size of $s_c = 64$ input samples. Raw data was generated randomly with a frame size of 128×512 pixels. The functional correctness of all correlators was ensured by comparing the individual output with the expected result of the test dataset, which was calculated in advance.

The code was run on two different platforms with eight virtual CPU cores each. One platform used a AMD *Piledriver* CPU (*FX-8320*), the other platform used an INTEL *Haswell* CPU (*i7-4770*) with 16 GiB DDR-3 memory each. See appendix A.2 for details on the system configurations

In case of the *Haswell*, hyper-threading (HT) is used to virtually double the amount of cores to eight. With two AVX units per physical core, each virtual core has access to a dedicated FMA enabled FPU. In case of the *Piledriver* microarchitecture, four modules are available with two cores each. Each physical CPU core contains a dedicated SSE FPU. To perform AVX operations, both FPUs in a module act jointly, actually halving the performance.

Influence of multi-threading

Figure 5.4 shows the performance of the different correlator implementations as a function of the number of threads. For both platforms, the performance grew approximately linearly with the number of real cores. For the *Haswell* architecture, the performance still increased for the HT-cores, but the slope was more shallow as these cores are no full cores. If the number of threads exceeds the number of virtual cores, the performance did not increase any further because the FPUs were effectively saturated. The same applied for the SSE implementation, as the the number of dedicated vector units is equivalent.

Although the eight cores of the AMD processor share only four AVX units, the performance surprisingly scaled linearly up until eight threads. When more than four threads were used, a performance decrease was expected as now two threads share a single FPU. At the same time the overall performance was significant lower than that of the SSE implementation (see Figure 5.6 for a detailed comparison of all implementations). Probably here the bandwidth to the caches was limiting the overall performance. For the SSE variant, that had access to a dedicated FPU per core, the course of the curve seemed to be similar to the *Haswell* platform. This variant also achieved a two fold higher performance than the AVX implementation on the same hardware. As expected, on both platforms the curves leveled up when the number of hardware resources are exhausted.

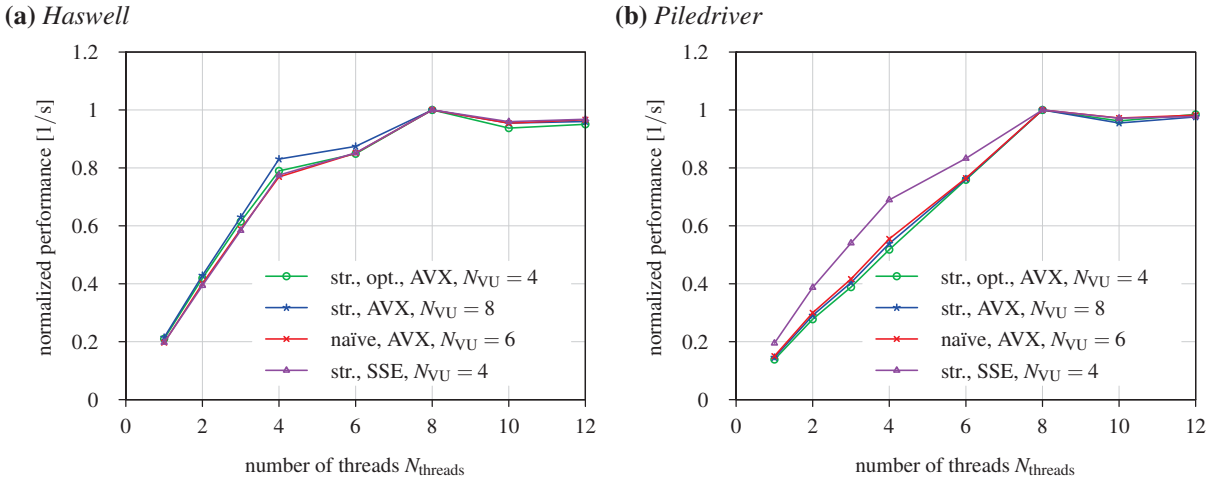


Figure 5.4.: Performance of the CPU correlator in relation to number of threads used. Data is shown for both evaluated CPUs and both SIMD instruction set extensions (AVX and SSE) and different implementations. The curves are normalized to $N_{\text{threads}} = 8$. *str.* denotes the 'streaming implementation', *opt.* a manually optimized implementation. The slope of the green curve in (a) is 0.19, and 0.12 in (b) for $N_{\text{threads}} = 1 \ldots 4$. Evaluations are based on a random input dataset with 240 000 frames.

Influence of the chunk size

In case of the streaming algorithm, the chunk size s_c is a parameter that allows for additional tuning of the correlator performance. For every block of each correlator a buffer for s_c samples is allocated. Therefore, lowering the chunk size has a significant influence on the memory footprint of the correlator. Depending on the chunk size, the memory blocks that are required to store output buffers may or may not fit into CPU caches. If they fit, this effectively lowers the bandwidth requirements for the external RAM. If s_c is increased, the correlator processes more consecutive values, thus requiring less context switches.

Figure 5.5 shows the influence of the chunk size on the runtime of the correlator for the example of the streaming algorithm. Tests showed, that for $s_c < 1024$ no significant decrease in performance was detectable. With regards to the memory consumption and also the runtime of the flushing procedure, a smaller chunk size is desirable. Due to these requirements and the findings above, $s_c = 64$ was chosen for all further evaluations.

Comparison of the different implementations on different platforms

Listing 2 and Listing 3 show the assembly language code of two lags in the linear correlator as generated by the *C* compiler for the implementation of the naïve and streaming algorithm. This code is reused for all blocks. The main difference is, apart from the amount of vertical unrolling, which is clearly visible in the number of consecutive FMA operation, the number of load/store instructions (`vmovps`) relative to the amount of math instructions (`vfmadd231ps`). In the second implementation, only two instead of four load/store instructions are used per lag. As only one block is processed at a time in the streaming algorithm, both, the input values and the delayed values are kept in the register file. In the implementation of the naïve algorithm only the input values resides in registers. For both implementations, the accumulators are held in memory.

In Listing 2, the ratio of load and store instructions on the total number of FMA instructions is $^8/_{12} = 0.66$ whereas in Listing 3 the ratio is $^4/_8 = 0.5$. Due to pipelining of the CPU, the performance of the streaming implementation is expected to be higher as the time spent waiting for memory can be bridged with arithmetic instructions. For both, the ratios of math-instructions on the total number of instructions

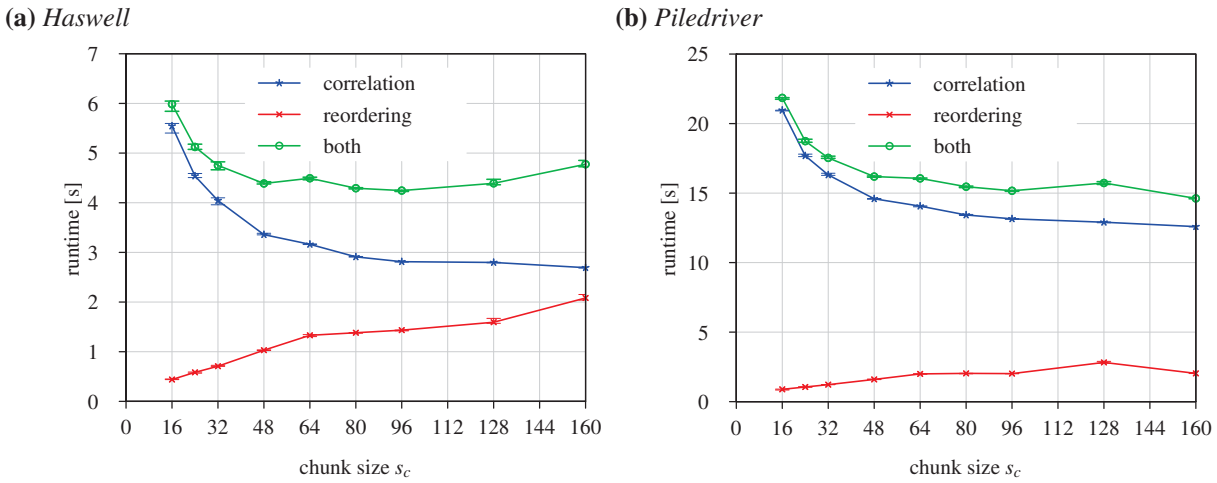


Figure 5.5.: Runtime of the CPU correlator as a function of the chunk size. Data is shown for both evaluated CPUs and the AVX instruction set extensions. The evaluation is based in the hand-optimized implementation of the streaming algorithm with $N_{\mathrm{VU}} = 4$. The runtime of the correlation only is shown in blue, time used for the reordering process is shown in red and the sum of both in green. Points indicate median values, error-bars minimum and maximum. Measurements were done using an input dataset of 720 000 frames filled with random data.

are identical: $^{8}/_{12} = {}^{12}/_{18} = 0.67$.

Figure 5.6 shows the actual performance of the different implementations. The implementation of the naïve algorithm is shown in red, the generic implementation of the streaming algorithm is shown in blue, and a hand-optimized version (*i.e.*, explicit loop unrolling of inner loops, manually pinned registers) of the same algorithm is shown in green. The AVX implementation of the naïve algorithm with $N_{\mathrm{VU}} = 6$ also uses manual register assignment.

It appeared, that the implementation of the naïve algorithm was always outperformed by the streaming correlator as expected from the ratio of math instructions. Manual loop unrolling or register allocation (hand-optimized implementation) had a significant impact on the achieved performance. In case of the AVX instruction set extension, the implementations behaved differently on the two evaluated platforms. For the SSE variants, the behavior in terms of relative performance was comparable.

Considering the size of the register file, vertical unrolling up to $N_{\mathrm{VU}} = 6$ is possible on both platforms without requiring to constantly exchange the registers with memory. Therefore, this was expected to be the most efficient implementation of a correlator for the naïve algorithm, but only applied for the AVX-based implementations. Accordingly, the implementations based on $N_{\mathrm{VU}} = 8$ were less efficient than the implementations based on $N_{\mathrm{VU}} = 4$, which was true except for the AVX-based implmentations on the AMD platform.

Runtimes were very reproducible, except for some outliers. For the INTEL platform, the jitter was typically less than 1 % around the median value. On the AMD platform, almost no jitter was measured. The performance of the correlator on both platforms dropped by up to 6 %, when a three times larger dataset was correlated (6 GiB file-size instead of 2 GiB). This effect is neither specific to the implementation of the streaming algorithm nor to the naïve algorithm. Therefore, the drop in performance might be related to the memory management of the OS.

Using the SSE implementation, about 50 % of the theoretical peak performance was reached on both platforms*. In case of the *Haswell* platform using AVX, about 40 % of the peak performance was reached. On the AMD platform, approximately 20 % of the theoretical peak performance was achieved, only half

*On the INTEL *Haswell* microarchitecture, the theoretical peak performance for both SSE and AVX differ by a factor of two (*cf.*, section 2.6).

of the performance of the SSE implementation. Even if the internal scheduling of the two FPUs per CPU module that are combined to a single AVX unit is not running optimally, a significant increase in performance should be visible for only four threads, as here an AVX unit is exclusively available per thread.

Double precision floating-point performance

As described in section 5.3.3, single precision floating-point values are not sufficient when a bleach correction is applied to the raw data prior to the correlation. Thus, the next wider data type,*i.e.*, double precision floating-point values, is required in such a case. With their double width, these values halve the number of entries in SSE or AVX vectors. Measurements showed that this also results in a two-fold decrease in performance (data not shown). An influence due to a less efficient input data conversion was not detectable (conversion of two half-words instead of a single word).

Influence of the ROI on the performance

To test the influence of the size of the detector or eventually the chosen region of interest (ROI), various frame sizes were tested. Figure 5.7 shows the results of a performance evaluation for different frame sizes. It can be seen, that the runtime scales linearly with the frame size. Therefore the frame size does not have a significant influence on the performance of the algorithm, which is expected since every pixel is processed independently (GUSTAFSON'S law, see section 2.5.1).

```
1        vmovaps 0x120(%rcx),%ymm13
2        vmovaps -0x58(%rsp),%ymm15
3        vfmadd213ps 0x140(%rcx),%ymm13,%ymm14
4        vfmadd231ps %ymm11,%ymm4,%ymm14
5        vfmadd231ps %ymm9,%ymm3,%ymm14
6        vfmadd231ps %ymm6,%ymm2,%ymm14
7        vfmadd231ps %ymm7,%ymm1,%ymm14
8        vfmadd231ps %ymm8,%ymm0,%ymm14
9        vmovaps %ymm14,0x140(%rcx)
10       vmovaps 0x1a0(%rcx),%ymm14
11       vfmadd213ps 0x180(%rcx),%ymm5,%ymm15
12       vfmadd231ps %ymm13,%ymm4,%ymm15
13       vfmadd231ps %ymm11,%ymm3,%ymm15
14       vfmadd231ps %ymm9,%ymm2,%ymm15
15       vfmadd231ps %ymm6,%ymm1,%ymm15
16       vfmadd231ps %ymm7,%ymm0,%ymm15
17       vmovaps %ymm15,0x180(%rcx)
18       vmovaps %ymm7,0x1a0(%rcx)
```

Listing 2: Assembly language representation of the implementation of the naïve correlation algorithm generated by the *C*-compiler for the INTEL *Haswell* CPU. This excerpt shows two out of eight lags in the basic linear correlator. Six FMA instructions are in a sequence, representing the amount of vertical unrolling, $N_{\mathrm{VU}} = 6$. The first FMA instruction (*e.g.*, line 3) also loads the contents of the accumulator into the register file.

```
1        vmovaps 0x88(%rsp),%ymm15
2        vfmadd231ps %ymm5,%ymm3,%ymm15
3        vfmadd231ps %ymm6,%ymm2,%ymm15
4        vfmadd231ps %ymm7,%ymm1,%ymm15
5        vfmadd231ps %ymm8,%ymm0,%ymm15
6        vmovaps %ymm15,0x88(%rsp)
7        vmovaps 0x68(%rsp),%ymm15
8        vfmadd231ps %ymm4,%ymm3,%ymm15
9        vfmadd231ps %ymm5,%ymm2,%ymm15
10       vfmadd231ps %ymm6,%ymm1,%ymm15
11       vfmadd231ps %ymm7,%ymm0,%ymm15
12       vmovaps %ymm15,0x68(%rsp)
```

Listing 3: Assembly language representation of the implementation of the streaming correlation algorithm generated by the *C*-compiler for the INTEL *Haswell* CPU. This excerpt shows two out of eight lags in the basic linear correlator. Four FMA instructions are in a sequence, representing the amount of vertical unrolling, $N_{\mathrm{VU}} = 4$. `ymm15` holds the accumulator. The `vmovps` instructions preceding and following each block of FMA instructions load and store the sum (*e.g.*, line 1 and 6). All other values are kept in the register file.

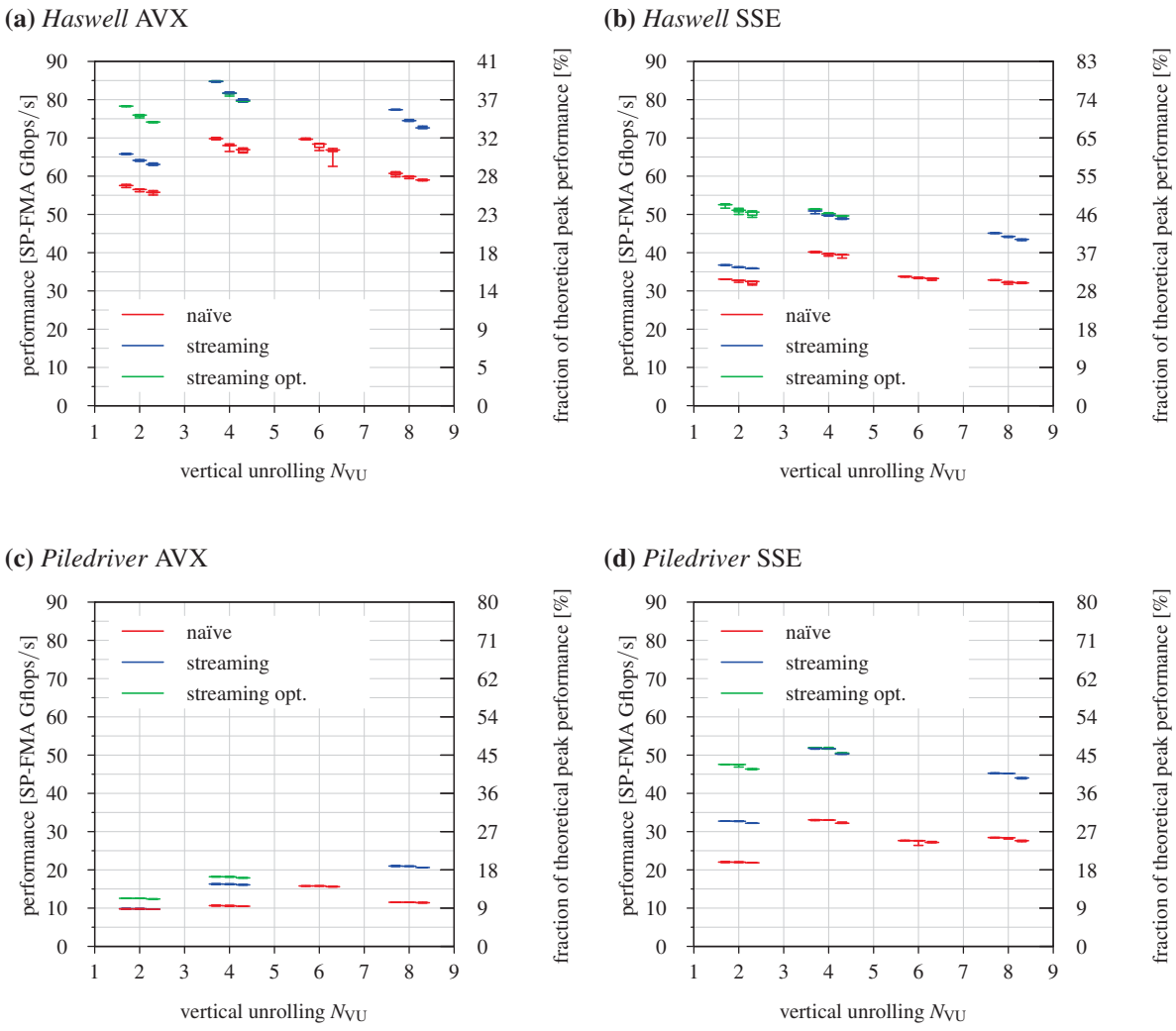


Figure 5.6.: Runtime of different implementations of the CPU correlator in relation to the vertical unrolling. Data is shown for both CPU platforms and both vector instruction extensions (AVX and SSE). The implementation of the naïve algorithm is shown in red, the generic implementation of the streaming algorithm is shown in blue, and the hand-optimized in green. Reordering of the raw data is not taken into account. For each dataset the three values from left to right represent a raw dataset of 240 000 frames, 480 000 frames, and 720 000 frames, respectively. The median is shown as a broader line. Real time performance of the correlator would be 164 Gflops/s. The theoretical peak performance was taken from Table 2.9.

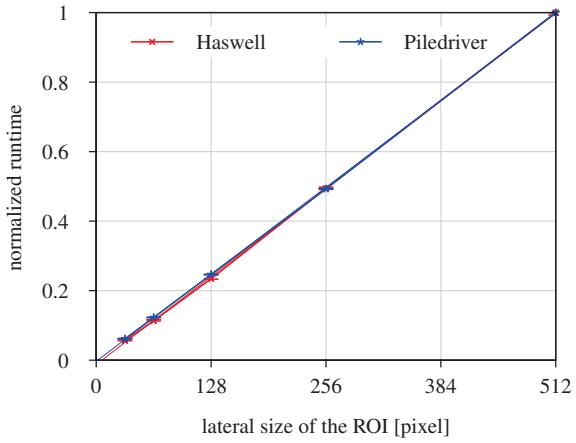


Figure 5.7.: Influence of the size of the ROI on the CPU correlator performance. The size of the ROI was set to $128 \times 1 \ldots 512$. For both platforms the slope of the curves is ~ 1. A dataset of 720 000 frames filled with random data was used for the performance measurements. Data is normalized to the full frame. The evaluation is based in the hand-optimized implementation of the streaming algorithm with $N_{\mathrm{VU}} = 4$.

5.3.6. Performance summary

Table 5.2 shows the maximum performance that was achieved for the CPU correlator on both CPU platforms, using the AVX and SSE instruction set extensions. The fact that on both platforms more than 40 % of the theoretical peak performance of the CPUs was achieved with the streaming algorithm, did not indicate that its implementation was heavily IO bound.

For the CPU implementation of the streaming algorithm, the theoretically required memory bandwidth exceeded the actual bandwidth by a factor of 4. Each correlator block processes $s_c = 64$ input values, so that the input and output fits into the caches. If then the next block can directly access this data from the cache, no access to the external memory is required.

Similar results regarding the performance were obtained using the *LinPack* [56] benchmark on the same *Haswell* CPU model. Using AVX and optimized libraries, a performance of 177 Gflops/s was achieved [110]. Therefore, it seems that the correlator reaches nearly the complete available performance.

For the correlation algorithm, the INTEL *Haswell* microarchitecture outperformed the AMD *Piledriver* microarchitecture if the CPU has the same number of virtual cores*. The larger number of FMA-FPUs and the wider interface to the caches on the INTEL chip proved advantageous, although the AMD CPU had a higher theoretical memory bandwidth to the external RAM.

	th. peak performance	SP-FMA performance AVX			SP-FMA performance SSE		
	[Gflops/s]	[Gflops/s]	relative	of real time	[Gflops/s]	relative	of real time
INTEL *i5-2520M* (*Haswell*)	217	85	0.39	0.52×	53	0.24	0.32×
AMD *FX-8320* (*Piledriver*)	112	21	0.19	0.14×	52	0.46	0.32×

Table 5.2.: Achieved maximum performance of the CPU correlator. Data is shown for the fastest implementation in each case (median value). Relative performance is shown in comparison to theoretical peak performance of the CPU for the AVX instruction set extension.

*Equating AMD *Piledriver* 'modules' with INTEL *Haswell* HT-cores.

5.3.7. Conclusion

Although the goal of real-time correlation could not be achieved on the evaluated CPUs, a significant reduction in correlation time compared to an existing implementation was possible. The fastest implementation reduced the total processing time by a factor of almost 4000× compared to the existing implementation (see Section 5.1.1).

On both CPU platforms, 40 % to 50 % of the theoretical peak performance was achieved, when the streaming algorithm was used. This is comparable to results achieved with for example the *LinPack* benchmark, which measures the practically achievable performance [110]. Interestingly, the SSE based implementation outperformed the AVX implementation on the AMD platform. Here possibly the shared FPU (one AVX FPU for two cores) lead to a decrease in performance.

Across all platforms, the streaming algorithm proved to be advantageous. Although using a larger memory footprint, this implementation was approximately 25 % faster than the implementation of the naïve algorithm. The main cause is the reduced fraction of load/store instructions and the beneficial use of the cache for buffers in between correlator blocks. Double blocks, as used for the GPU correlator, were not tested due to the limited size of the CPU register file.

5.3.8. Outlook

As mentioned before, the real-time requirements of the CPU correlator can be achieved by simply adding CPU cores to the system. This can either be done by physically adding another CPU to the computer or by using a CPU model that comprises more cores. Here the INTEL *Haswell 5960X* with eight physical cores should be sufficient to achieve real-time performance.

The complex interplay of optimizations, that were done by hand or by the compiler, and the highly complex CPU make it difficult to predict the performance of each individual implementation. In order to select a specific correlator implementation for a given computer architecture, all different parameters have to be tested (*i.e.*, vertical-unrolling, chunk size, number of threads, AVX or SSE, etc.). If the correlator should be shipped as a package with an evaluation tool, *e.g.*, *QuickFit3*, auto-detection of the best implementation should be done on every new hardware to optimize the correlation performance.

As already shown for the streaming correlator, different implementations of linear correlator blocks can be plugged together simply. Here, the first stage differs much from the others. This concept of plugging together different blocks also allows for using different data types (*i.e.*, double precision floating-point) in later stage without a huge impact on the overall performance. Additionally, it is possible to use a wider data type for the accumulators only.

With the INTEL *Xeon Phi* co-processor, a single instruction for mask loading was introduced. This can be used to convert raw binary input data to floating-point vectors (`_mm512_int2mask`), and should reduce the number of instructions used for conversion to two. Furthermore this architecture (*Knights Landing*) uses double width SIMD units (AVX-512 with 512 bit instead of 256 bit) and comprises more than 50 individual CPU cores. With a performance above 1 Pflops/s, this is an interesting architecture for a future high-performance correlator.

5.4. GPU correlator

This section describes the design and implementation of a multiple-τ correlator on a GPU. The implementation is based on NVIDIA's CUDA API. An introduction to GPU computing using compute unified device architecture (CUDA) is given in section 2.7. Table 5.3 lists the software that was used for the development of the correlator.

software	version
NVIDIA GPU drivers	331.20
CUDA toolkit	5.5.22
CUDA profiler (*nvprof*)	5.5
CUDA visual profiler (*nvvp*)	5.5.0

Table 5.3.: Software used for the development of the GPU correlator.

5.4.1. Hardware platform

All evaluations were carried out on the *B040-SPIM2* computer (*cf.* section A.2) with an EVGA *GTX 780Ti* GPU. Its specifications are shown in Table 2.8. The GPU is based on the *GK110 Kepler* chip from NVIDIA.

For the development of the correlator, a consumer-grade GPU was used. It performs close to the current NVIDIA's single-chip top model GPU *Tesla K40*, which is based on the same chip. However, the *GTX 780Ti* variant has a slightly higher base clock rate and consequently a higher single precision performance. It is only outperformed by the TESLA *K80* which is based on two *GK210* chips. Being a consumer graphics card, the *GTX 780Ti* lacks features like error correcting memory or a larger graphics RAM.

An important limitation of this test system was, that only a PCIe 2.0 16× link could be used between CPU and GPU. This limits the bandwidth to a theoretical value of 7.8 GiB/s.

All evaluations were done using compute model 2.0. Although the hardware is compatible to compute model 3.5, tests showed that the register usage was lower and the overall performance was significantly better due to a higher occupancy when using compute model 2.0.

5.4.2. Implementation of the GPU correlator

The GPU correlator is based on the streaming variant of the multiple-τ algorithm (see section 5.2.3). Each single linear block of the correlator is represented by a compute kernel which is executed on the GPU. Therefore, several of these compute kernels have to be executed serially in order to calculate all lags (*i.e.*, all blocks) of the complete correlator. The order in which these compute kernels are processed, as well as the scheduling of the data transfers between the CPU and the GPU is done on the host CPU. To not limit their performance, the compute kernels had to be kept as simple as possible. Especially unpredictable branching that might lead to a stalling of a part of the thread group ("warp") on the GPU had to be avoided (*cf.*, section 2.7).

When a compute kernel is started, it first loads the state (called "context" in the following) of the current linear correlator block from GPU memory to the register file. Then it processes a fixed number of input frames in a *for* loop and finally writes back the altered context. Each compute kernel comprises 65 536 GPU-threads, where each thread processes data from one pixel. The scheduling of the parallel execution is done by the GPU itself. The threads are organized into 256 blocks with 256 GPU-threads each. In this implementation, all compute kernels (or blocks) are processed serially, which implies the possibility to run other processing tasks in parallel to the ongoing correlation on the GPU. This way, also halting-states (*e.g.*, when a thread waits for a memory access) can be exploited to perform additional compute intensive tasks, such as a fit (*cf.*, SIMT, section 2.5.1).

The *GTX 780Ti* contains 3 GiB dedicated graphics RAM. Two thirds (2 GiB) are used to store input/output-data of the linear correlator blocks (compute kernels). The remaining 1 GiB is used to hold the contexts of all blocks and in addition the raw data that is processed by the first block, the so-called 'front end'. To save bandwidth, the raw binary data from the CHSPAD detection system (see section 4.2 for details on the design) is transferred to the GPU without further processing. Since the correlation

algorithm operates on single precision floating-point values, raw image data has to be converted before the correlation. This task is performed by the front end, too.

In total, three different compute kernels (*i.e.*, blocks of the correlator) are used: The front end compute kernel performs the data conversion and processing of the first block. A 'generic' correlator compute kernel is used for all other blocks except the last one. Finally, the 'last-stage' kernel is identical to the generic correlator but does not generate any output for further blocks. That way, its performance was significantly improved (by approx. 22 %). Each compute kernel is started with five parameters: the current block level, the source of the input data, the destination of the output data, the storage address for the context and the number of frames to process. The context comprises the state of the accumulators and the delayed values, which is also the data that is finally returned by the algorithm. In case of the front end compute kernel, the storage address for the monitor channels has to be passed to the compute kernel in addition to the other five parameters.

Figure 5.8a shows the naïve implementation of the scheduling of the compute kernels and the memory transfers without use of asynchronous data transfers. To allow transferring of new chunks of raw data to the device in parallel to the ongoing processing of the compute kernels, two CUDA streams are executed simultaneously (see Figure 5.8c). The first CUDA-stream alternatingly handles the front end correlator and the CPU-to-GPU data transfer. A second CUDA-stream exclusively handles the scheduling of higher order correlator blocks. Both streams are synchronized when the front end correlator exits and again after the higher order blocks have been processed. This way, the data transfer is done in parallel to the processing of higher order compute kernels.

For the actual implementation of this algorithm the standard configuration of 16 blocks per correlator with 8 lags each was used. Figure 5.9 shows the streaming of input data for the first three blocks, together with the intermediate storage of input and output data of the different blocks: The first correlator is run in a way that its output complete fills the available storage. The data is then consumed by the second correlator block. Here the data is reduced by a factor of two and written to the first or second half of the output storage. If the same correlator block is run again, the other half is filled. After that, the next correlator can be run again consuming a completely filled storage.

When 2 GiB are used for storing input/output data, 128 MiB are available in each of the 16 blocks. To completely fill this output storage space of each block, 512 frames* of single precision floating-point numbers have to be processed by the block†. As shown in Figure 5.8b, the data transfer from the CPU to the GPU may take longer than execution of a compute kernel for a higher order block. This leads to a stall of the entire correlator, as the input data is not ready to be processed. To overcome this, instead of running all compute kernels with the same number of input frames (256 frames, Figure 5.8a), the front end compute kernel gets twice the amount of input frames, and two higher order blocks are executed during the transfer. This strategy overcomes the stalling, but doubles the amount of raw data that has to be transferred to the GPU in each copy operation to 4 MiB for 512 frames. The scheduling in this mode is illustrated in Figure 5.8c. Now both CUDA streams run compute kernels and data transfers, and no time is spend waiting for memory transfers to complete.

GPU data structures. The context of each correlator consists of the contents of the accumulators and the delay elements. It has a size of 4 MiB for each block. To improve the load/store performance, the data is organized thread-major, so that neighboring threads access adjacent data. This can be visualized as follows:

$$\text{thread id} : \underbrace{\underbrace{\underbrace{0,1,\ldots,N-1}_{\text{first lag}},\underbrace{0,1,\ldots,N-1}_{\text{second lag}},\ldots,\underbrace{0,1,\ldots,N-1}_{\text{last lag}}}_{\text{accumulators}},\underbrace{\underbrace{0,1,\ldots,N-1}_{\text{first lag}},\ldots,\underbrace{0,1,\ldots,N-1}_{\text{last lag}}}_{\text{delay elements}}}.$$

*Size of a frame: 512×128 pixels

†$N_{\text{frames}} \cdot \text{xres} \cdot \text{yres} \cdot N_{\text{Bytes per float}} = 2 \cdot \frac{1}{2} \cdot 512 \cdot 128 \cdot 512 \cdot 4\,\text{B} = 128\,\text{MiB}$

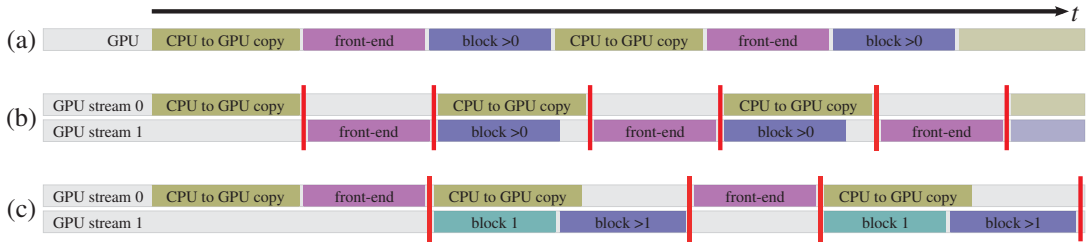


Figure 5.8.: Parallelization of the GPU correlator execution and data transfer with two CUDA streams. In (a) a linear implementation is shown. First, data is transferred to the GPU (yellow). Then the first block (the front end compute kernel, violet) is executed. After that the higher order block compute kernel (dark blue) is run and so forth. In (b), data transfers are done in parallel to the higher order block. If a data transfer takes longer than the runtime of a compute kernel, the GPU is idle. The "initial" implementation is shown in (c). Here the front end processes twice the amount of frames so that the second block (turquoise) has to be run immediately after, prior to the execution of a higher order block. The "optimized" implementation using "double blocks" uses the scheme shown in (b). Synchronization barriers between threads are shown in red.

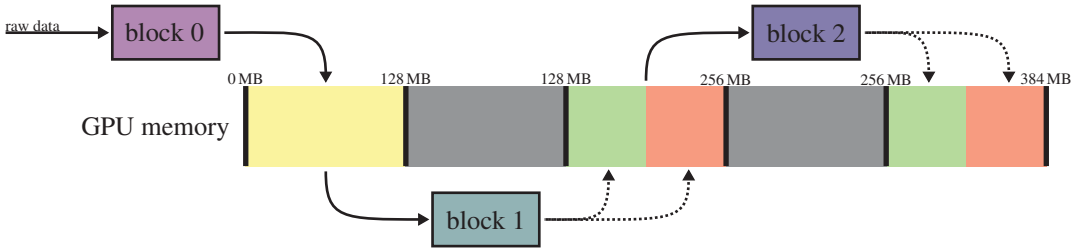


Figure 5.9.: Schematic drawing of the streaming algorithm used for the GPU correlator. The front end (block 0, violet) processes twice the amount of frames than the other blocks. Every other block takes a full 128 MiB dataset as input which is reduced to 64 MiB at its output. This output is either written to the first or second half of the next blocks' input. Coloring is in accordance with Figure 5.8c.

In contrast, a naïve implementation would be to use the lag-major memory layout:

$$\text{lag id} : \underbrace{\underbrace{0,1,\ldots,N-1}_{\text{first correlator}},\underbrace{0,1,\ldots,N-1}_{\text{second correlator}},\ldots,\underbrace{0,1,\ldots,N-1}_{\text{last correlator}}}_{\text{accumulators}},\underbrace{\underbrace{0,1,\ldots,N-1}_{\text{first correlator}},\ldots,\underbrace{0,1,\ldots,N-1}_{\text{last correlator}}}_{\text{delay elements}}.$$

This layout is easier to debug but leads to an about 30 % lower performance and was therefore used only during the development of the correlator. The reason for this is, that neighboring threads access adjacent data words, which is much more efficient in the thread-major layout (coalescing access). The same applies for the buffers used input and output data of the correlator blocks. Additionally, the global (un-delayed) values and the local (delayed) values are stored in an alternating manner.

By packing several data structures into the same memory block, for example the accumulators and delay elements or the local and global values, fewer registers are required for the access of the data. Only the base pointer needs to be held in a register. The actual position of the values relative to this base address is then derived from the index of the thread. Reducing the number of registers can in turn increase the occupancy of the compute kernel and thus increase the overall performance.

Correlator data type

The GPU correlator was designed for single precision floating-point data only. In principle, the correlator could also be implemented to use double precision, but then the performance drops significantly (24× less) due to the limitations of the GPU (artificially reduced double precision floating-point performance in consumer-grade variants). However, the modular structure of the compute kernels would allow to use double precision floating-point values for a subset of correlator blocks. For example only higher order

vert. unrolling N_{VU}	avg. compute kernel exec. time [ms]	#registers	occupancy [%]	performance [Gflops/s (SP FMA)]
32	6.8	148	12.5	178
16	1.5	103	25	422
8	1.5	98	25	418
4	1.5	101	25	419

Table 5.4.: Influence of the vertical unrolling on the performance of the GPU correlator. Results are shown for the generic compute kernel of the optimized implementation.

blocks can be implemented using double precision floating-point values as here an increased precision can be beneficial. As each block is executed half as often as its predecessor, a longer execution time of higher order blocks will not have a huge impact on the overall performance.

Verification

The results of the GPU correlator were verified against the CPU correlator. Both implementations (GPU and CPU) yielded identical values.

Flushing

When all raw input values have been processed, the correlator has to be flushed in order to propagate all values through all lags of the entire correlator (*cf.*, section 5.3.4). To reduce the time that has to be spent for this operation, the flushing algorithm has further been optimized. Instead of running the entire correlator for a certain number of input values until the last block has been executed a sufficient number of times, each block is executed just as often as needed until it is flushed with zeros. When the flushing starts, the normal scheduling of the blocks in the correlator is continued. After the first block has been run L times, its execution is blocked as now the output is completely zeroed. The same applies for the second block: After the execution of the first block is stopped, it is only executed further L times, and so forth. The entire procedure took about 0.2 s instead of 0.5 s for this improved algorithm.

Vertical unrolling

Vertical unrolling, described in section 5.2.8, combines several adjacent time steps of the correlation algorithm into a single one. Thus, it reduces the number of required load and store operations, and the memory-bandwidth requirements can be lowered. Furthermore, by using more registers, vertical unrolling makes better use of GPU architecture. In case of the CPU correlator, this approach already proved to be advantageous and a significant performance increase could be achieved (approximately 30 % by increasing $N_{VU} = 2$ to $N_{VU} = 4$).

In case of the GPU, the entire context of a correlator block is held in the register file. The context is loaded once when a compute kernel is started, and written back to GPU memory when a compute kernel exits. As the register file of the GPU in comparison to the CPU is much larger larger a higher degree of vertical unrolling is possible. Thus allowing compute kernels to process more time-steps of the correlation algorithm simultaneously.

During the runtime of a compute , the full memory bandwidth is used to load and store the intensity values that pass through the correlator block. As shown in Table 5.4), vertical unrolling has only a minor influence on the compute kernel performance, as memory accesses are limiting the overall performance Only when the register count reduces the occupancy, a significant performance drop is detectable (*e.g.*, $N_{VU} = 32$).

Contrary to that, vertical unrolling is important when more than one block is combined into a single compute kernel. As each following block consumes two adjacent input values simultaneously to generate a single output value, the preceding block must consume four consecutive values to provide these two, and so forth. Thus, the vertical unrolling of each preceding block doubles, so that:

$$N_{\text{combined blocks}} \leq \log_2 (N_{\text{VU}}) . \tag{5.4}$$

At the same time, the number of registers is increased by the size of a block's context, which is $2 \cdot N_{\text{lags}}$. This in turn can lead to a reduced occupancy.

Further optimizations

As indicated in the last paragraph, the performance of the correlator is mainly limited by the bandwidth of the GPU's memory. To effectively decrease the amount of data transfers, a new compute kernel was designed (so called 'optimized' implementation) that executes two blocks in a single compute kernel. Then the output of the first block is directly consumed by the second block and does not have to be stored in GPU memory in between. This is most efficient, if the level of vertical unrolling of the first block is twice as high as the second block: Four input values of the first block are combined to two input values of the second block which is then reduced to a single output value of the compute kernel. In total, the required amount of data transfers is almost halved ($^5/_9 = 0.56$). The resulting execution scheme is comparable to the one shown in Figure 5.8b).

In case of the front end this is extremely efficient as the input data is neglectable in comparison to the output. Here, the execution time could be reduced by about 15 %, whereas at the same time the amount of processed blocks is doubled. For higher order blocks the reduction is about 8 % in processing time, but in this time 1.5-times more operations are executed, *i.e.*, a full block and a half of the following block are executed. As a side effect, also the storage for input and output is reduced by a factor of two as now every second correlator does not produce any output. These 'double-blocks' require a new scheduling scheme, because data reduction is fourfold in comparison to the twofold reduction of the original scheme. Instead of every two, the next block only has to be executed every four cycles. A simple approach to this scheduling is to always execute the lowest order block that has a completely filled input buffer. As this scheduling is done on the CPU, it has no influence on the code and execution time of the compute kernels.

5.4.3. Performance analysis

Maximum performance considerations

The single precision peak performance of the GPU (2.5 Tflops/s (FMA)) would theoretically be sufficient to calculate the correlation estimates for all pixels of the CHSPAD in about $1/15$-th of the time that is required for the data acquisition (*cf.* section 5.2.6). This limit can only be reached, when memory operations are not taken into account and if the code has 100 % occupancy. Also, the CPU-GPU connection (here via a PCIe-2.0 bus) has a huge influence on the overall correlator performance. In the setup described above, the bandwidth was 5.0 GiB/s which already limits the speedup to $\sim 4.2\times$ above real time, if only CPU-to-GPU data transfers are considered. Finally, the interface to the GPU memory is also important. In the implementation described above, the front end compute kernel reads 8 MiB and writes 132 MiB (512 frames) and the higher order blocks read 132 MiB and write 68 MiB (256 frames). In each cycle, the first block is executed once and the others are executed twice which results in 540 MiB of data to be transferred (full correlation of 512 frames). At the given peak memory bandwidth, this will therefore limit the overall speedup to $\sim 2.7\times$.

Influence of the chunk size

As seen for the CPU correlator, the number of frames processed by a correlator block in a single run, the so called chunk size, has an impact on the overall performance. Table 5.5 shows the results from a performance evaluation for different chunk size configurations. It shows that the performance maximum is reached for the standard configuration (256 MiB), which requires 2 GiB of GPU memory. Even if the chunk size is doubled again (512 MiB) and less less blocks re processed (8 instead of 16), the performance was not further increased. Overall the performance is not very sensitive to the chunk size, even if it is reduced by a factor of four (drop of approximately 8 %). However, this configuration was much more efficient in terms of memory consumption as only 0.5 GiB of GPU memory are required for the correlation. Thus, additional memory can be freed for other tasks on the GPU, if a minor drop in correlator performance is acceptable.

Performance of the correlator

The performance of the correlator was tested extensively using randomly generated datasets. To measure the overall runtime of the correlator, CUDA events were used. An overview of the results for the different implementations is given in Table 5.6.

The initial implementation, which is based on single blocks, reached an overall speedup of 1.4× over real time, which means an instruction throughput of 229 Gflops/s (FMA). To further increase the performance towards the PCIe bandwidth limit, the additional steps described in the last section were implemented. This optimized implementation reached a speedup of 2.6× over real time, almost doubling the performance of the initial implementation. As memory transfers from CPU to GPU were not fully parallelized to the ongoing correlation (see Figure 5.8b). Hence, if only the time required by the compute kernel is taken into account, *i.e.*, assuming a fully parallelized data transfer, one could reach a performance of approximately 560 Gflops/s (speedup of 3.4× over real time).

Performance evaluation of the compute kernels

A full correlator is made up of at least two compute kernels. The front end handles data conversion and processes the first block(s) of the correlator. Further blocks are then handled by a second 'generic' compute kernel. To further optimize the performance a third compute kernel can be used for the last block, which omits any output.

In the following, a more detailed evaluation of the individual compute kernels is given. The performance measures are based on the output of the NVIDIA profiler *nvprof*. Table 5.7 summarizes the results of the performance evaluations of the different compute kernels.

The measurements show, that a maximum memory bandwidth of approximately 250 GiB/s can be reached. This seems to be the upper limit of the proposed access pattern for the correlator and is roughly 75 % of the peak performance of the memory interface.

chunk size	number of blocks B	memory requirements [GiB]	performance [Gflops/s (SP FMA)]	normalized performance
512 MiB	8	2	416	0.99
256 MiB	16	2	419	1
128 MiB	16	1	410	0.98
64 MiB	16	0.5	385	0.92
32 MiB	16	0.3	322	0.77

Table 5.5.: Performance of the GPU correlator in relation to the chunk size per correlator block.

Implementation	speed-up over real time	performance (SP FMA) [Gflops/s]
initial implementation (single block)	1.4×	229
optimized implementation (double-blocks)	2.6×	419
fully parallelized opt. implementation‡	≈ 3.4×	≈ 560

Table 5.6.: Performance of the GPU correlator. In the initial implementation, per compute kernel a single block is evaluated. The optimized implementation uses a single compute kernel per two linear correlator blocks. ‡: Assumption based on compute kernel runtime only.

For all compute kernels except for the optimized front end, this was limiting the total performance. Even a higher occupancy (optimized generic compute kernel), did not improve performance, as the threads are mostly waiting for memory transfers to complete. In case of the optimized front end, reducing the dependency on memory transfers by combining two correlator blocks into a single compute kernel, doubles the performance of the compute kernel to almost 35 % of the peak performance of the GPU.

5.4.4. Conclusion

The GPU-based correlator achieved a performance of about 420 Gflops/s, a speed of 2.6× above real time. With that, the correlator outperformed the CPU and the FPGA platform, as well as every other implementation proposed so far (section 3.2).

The correlator achieved 70 % of the peak memory bandwidth, which seemed to be the upper limit on the GPU. In case of the front end compute kernel, with less requirements to the memory bandwidth, a performance above 850 Gflops/s was reached, which indicated that the algorithm is mostly IO bound.

If only floating-point operations related to the actual correlation are considered, a performance of up to 560 Gflops/s was achieved. This corresponds to 22 % of the theoretical peak performance of the GPU, or a speed of 3.4× above real time.

5.4.5. Outlook

As the GPU correlator over-fulfilled real-time performance, it should be possible to at least calculate two additional cross-correlation estimates per pixel. If then three correlators ran in parallel, the chunk size would have to be reduced. However, this can be done with only a minor drop in performance.

When running as a real-time correlator, the unexploited performance could also be used to calculate additional monitor channels. Currently only the front end counts the number of incoming photons for each pixel. It was shown, that the normalization of the correlation function estimates benefits if monitor channels are employed per correlator block.

compute kernel	#frames	runtime	RAM throughput			performance	#registers	occup.
			read	write	combined	(SP FMA)		
		[ms]	[GiB/s]	[GiB/s]	[GiB/s]	[Gflops/s]		[%]
initial impl., front end	512	0.64	14.0	217.3	231.3	419	53	50
initial impl., generic block	256	0.82	168.1	86.6	254.7	163	61	50
opt. impl., front end	1024	0.94	18.2	152.4	170.6	858	64	50
opt. impl., generic block	512	1.48	189.2	51.6	240.8	272	103	25

Table 5.7.: Performance of the compute kernels of the GPU correlator. Data is based on the metrics *dram_read_throughput*, *dram_write_throughput*, *flops_sp_fma* as well as the execution time that were evaluated using *nvprof*. Number of registers and occupancy was determined using *nvvp*. The optimized implementation (opt. impl.) is based on double-blocks. *occup.* refers to occupancy.

5.5. FPGA correlator

This section describes the implementation of an FPGA correlator. The correlator was designed as an integral part of the *RADHARD2* readout-system with a frame rate of 100 kHz and full frames (32×32 pixels). Its operation is based on the algorithm described in section 5.2.2. A single DSP cell combined with a dedicated memory block (BRAM) forms the basis to calculate all lags of a linear block as well as multiple blocks. The hardware is based on the *LASP* development board (see appendix B.1 for details on this board), featuring two XILINX VIRTEX-II *XC2VP40* FPGAs.

Even with these relatively old FPGAs, the correlator achieved real-time performance. The readout-system including correlator were successfully used in the SPIM-FCS setup. Parts of this implementation have already been published in Ref. [155].

5.5.1. Hardware platform

For the implementation, two XILINX VIRTEX-II *XC2VP40* FPGAs were available on the platform. Since one of them was mainly used for the readout, the correlator was implemented in the second FPGA. Both FPGAs are connected on the board via a parallel bus. Table 5.8 gives an overview of the resources that are available in such an FPGA and compares them to a more recent VIRTEX 6 FPGA.

FPGA design tools

Table 5.9 summarizes the development tools used for the design of the FPGA correlator. The support for VIRTEX-II FPGAs was dropped in newer design tools from XILINX. Therefore, the latest version featuring support for the VIRTEX-II was used.

5.5.2. Implementation

The design of the FPGA correlator is based on hand-written VHDL code (Very High Speed Integrated Circuit Hardware Description Language). The basic design goal was to calculate the correlation function estimates of all 1024 pixels of the *RADHARD2* SPAD array in real time. At that stage, the readout controller of the *RADHARD2* was capable to read 100 000 fps. Due to limitations of the available memory, the correlator processes 14 blocks with 8 lags each, spanning five orders of magnitude (10 µs to 1.23 s). Most parts of the correlator except for the required data reordering (see blow) were implemented into a single XILINX *XC2VP40* FPGA. Although the correlator was used for an autocorrelation analysis only, its generic design allows for cross-correlation, too.

The correlator core is mainly based on the MAC operation, which can be implemented using DSP slices. Thus, their speed determines the maximum performance of the whole correlator. The VIRTEX-II used in here contains 192 digital signal processor (DSP) slices that can be operated at a maximum frequency of 147 MHz [247]. The chip also contains the same number of BRAMs. Since always one DSP and one BRAM share some hardware resources, both cannot be used at full data width at the same

resource	XILINX VIRTEX-II *XC2VP40*	XILINX VIRTEX-6 *XC6VLX240T*
registers	39*k*	301*k*
LUT	19*k*	151*k*
BRAM	192 (3.4 Mbit in total)	832 (15.0 Mbit in total)
multipliers	192 (18x18)	768 (25x18)

Table 5.8.: Overview of the resources of the XILINX VIRTEX-II *XC2VP40* and the VIRTEX-6 *XC6VLX240T* FPGA. Here only resources that play an integral role in the correlator design are mentioned. The VIRTEX-II is used on the *LASP* board and the VIRTEX 6 on the XILINX *ML605* development board.

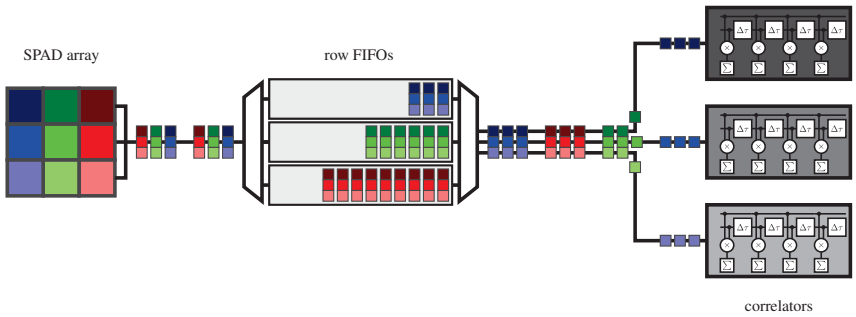


Figure 5.10.: Schematic of the data reordering and context switching of the FPGA correlator. As an example a 3×3 SPAD array is used.

time. If all DSPs work at their highest data rate, a peak performance of 28 Gflops/s is reached on this VIRTEX-II. If (as it will be done in the implementation of the correlator) each DSP is combined with at least one BRAM, the theoretical peak performance is reduced to 14 Gflops/s due to the resource sharing between the two coupled units *. For comparison, a more recent VIRTEX 6 *XC6VSX475T* FPGA has 2016 DSPs that can operate with up to 600 MHz, which gives a theoretical peak performance of 1210 Gflops/s [250, 252].

To maximize the overall performance of an FPGA-based correlator, as many DSP slices as possible have to be used. In principle, multipliers can be implemented in the fabric, too, but reach a comparable performance for small widths only and they would require a large amount of logic cells. Therefore such multipliers can only be used for the first linear blocks of a correlator.

Due to the limited number of 192 DSPs, in comparison to the requirement of 1024 correlators, the only possibility is to reuse the hardware to compute the correlation estimates for all pixels and share resources. On top of that, the overall amount of available memory is not sufficient to store an entire dataset from a single measurement on the board ($\sim$ 12 MiB/s or 720 MiB for a typical measurement of one minute). Therefore, processing all pixels one after the other at the end of a measurement is also no option. Due to all these limitations, a design was chosen, which reuses a single correlator to process multiple pixels. To do so, a fast pixel-context switch is necessary, which completely reconfigures the correlator, stores the intermediate results of the current pixel to external memory, and loads the data for a new pixel. Such a switching also requires a resorting of the data stream from the sensor into chunks that group pixels which are processed in parallel. That way, the correlators can process as many values as possible of the same row of the SPAD array continuously. When the operation on a specific row is finished, a context switch is performed, and the next row is processed for a certain time, and so forth. Figure 5.10 shows this principle of data reordering, that happens in parallel to the ongoing data acquisition (see Figure 4.2 for a schematic of the whole system). As raw data is read row-wise from the *RADHARD2* SPAD array in lines of 32 pixels, 32 identical correlators were implemented in parallel. To cover all 1024 pixels of the SPAD array, each single correlator has to process 32 pixels.

To allow fast switching of different linear blocks in a single correlator or between different pixels, the delay registers in the delayed data path (local values) in between the multipliers must be implemented in

product	version
SYNOPSYS (Mountain View, California, U.S.) *Synplify Pro* (synthesis)	D-2010.03
XILINX (San Jose, California, U.S.) *ISE* (implementation)	10.1.03
MENTOR GRAPHICS (Wilsonville, Oregon, U.S.) *ModelSim* (simulation)	10.1c

Table 5.9.: Software used for the development of the FPGA correlator.

*Since the VIRTEX 4 generation, both units work independently.

BRAMs. A context switch of a block level then becomes a change in the base address of the memory location. The order of blocks is determined by the block scheduling algorithm (see section 5.2.4 for details). The same applies for the sums in the accumulators, which will be returned as the final result of correlation. For each lag, these two data words (accumulator and delayed value) are combined into a 48 bit word which does not fit into a single BRAM which is 32 bit wide. Therefore, at least two BRAMs have to be used in each single correlator lag. In principle, resource sharing between different correlators is possible so that only 1.5 BRAMs are occupied. As the number of BRAMs and DSPs is the same, the available memory blocks are limiting the overall correlator design. In case of the VIRTEX-II, 192 BRAMs are available and by using 1.5 BRAMs for each correlator lag, one could implement up to 64 lags in parallel. If only one physically existing lag is used for a full correlator, at least 16 pixels need to share the same hardware to process 1024 pixels in parallel.

The context of a single pixel consist of 128 word of 48 bit, or 0.8 MiB for all pixels. This is about twice the amount of the resources available on the FPGA (*cf.* Table 5.8). To overcome this limitation, only the current context is loaded to the FPGA on-chip memory, whereas unneeded contexts are stored in external SRAM. A double buffering strategy is employed to keep the time required for context switching at a minimum: One part of the BRAMs attached to each correlator holds the current context while the other part holds the context of the prior/next block. During the processing of a certain row of the SPAD array, the currently unused context is exchanged with the next needed context from the external memory. When the processing of the current context is done, the buffers are switched. As contexts are constantly exchanged with external memory, this stream of data is directly output via the USB interface to allow the operator of the microscope following correlation.

A further limiting factor of the FPGA was the number of ports offered by the BRAMs: Each memory block has only two ports for reading and writing. If data has to be exchanged constantly, only one port is left for the actual correlation operations. In addition, each port can only be used to either read or write during a single clock cycle, which effectively halves the performance, as two clock cycles become necessary for processing one lag of a correlator.

Internally, each correlator uses a 16 bit wide data path for the delayed and un-delayed values as well as for the multipliers. The accumulators are 32 bit wide, but especially in the lags for large delay times they can still overflow frequently. These overflows are followed by a wrap-around in hardware, which is detected and corrected for later in the read-out software. This is possible, because the USB-interface does not only transfer the final results, but also some of the intermediates during the correlation. In fact, the USB-interface constantly streams intermediate results to the host that are first stored and then later mangled into final, normalized correlation curves.

The main clock is provided by the USB interface running at 48 MHz. The correlator core itself runs at 144 MHz, the threefold frequency. Since the MAC operation is simple to implement with the help of DSP cells, the data housekeeping was the more demanding task. Finally, the results of the correlation analysis are transferred to the host computer via a dedicated USB 2.0 link using the same frame format as for the raw data.

After the processing of all input values, the correlators have to be flushed to allow all values the propagation through the entire correlator. This is done according to the scheme described in section 5.3.4.

Single correlator design

Figure 5.11 gives an overview of two out of 32 correlators. A single correlator consists of a DSP cell used for the MAC operation and two attached BRAMs. One BRAM holds the 32 bit accumulators, the other one holds the 16 bit wide contents of the delay elements. As multipliers and BRAMs share resources (see above), a single correlator occupies two BRAMs and two multipliers. For the correlator, only one of the two memory ports of the BRAMs is available. The second port is used for context exchanging the contexts with external memory.

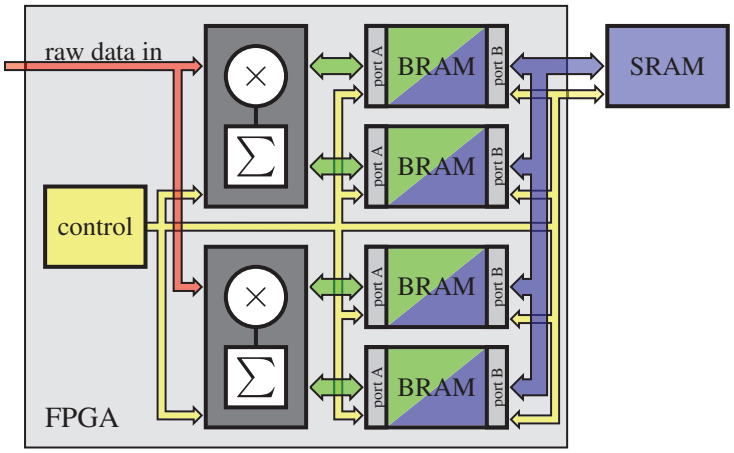


Figure 5.11.: Schematic drawing of the FPGA correlator. Only two of the 32 correlators are shown. Each correlator (dark gray) comprises a single lag which is re-used for the whole correlator and multiple pixels. Each correlator uses two attached BRAMs. The current context is shown in green. In parallel to the ongoing correlation, a second context is stored/loaded from external SRAM (blue) using the second memory port of the BRAMs. The whole operation in controlled by the controlling circuit (yellow) which also contains the block scheduler and memory exchange logic. Raw data (red) is captured and re-ordered by the first FPGA on the *LASP* board.

clock cycle c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
lag 0	L	W	M	A	S				L	W	M	A	S				lag 4
lag 1		L	W	M	A	S				L	W	M	A	S			lag 5
lag 2			L	W	M	A	S				L	W	M	A	S		lag 6
lag 3				L	W	M	A	S				L	W	M	A	S	lag 7

Table 5.10.: Inner core of the FPGA correlator: interleaved pipeline scheme for eight lags. As only one port of the BRAM is available, only a single **L**oad or **S**tore operation can occur per cycle. Further cycles are **W**ait for memory completion, **M**ultiply and **A**ccumulate.

position in context	1st BRAM 16 bit	2nd BRAM 32 bit
0…111	raw accumulator values $g_{\tau_{b,l}}$	delay register $I^d_{b,l}$
112	-	current cycle counter c
113…125	-	intermediate results for handover between blocks $I_{b,0}$
126	-	un-delayed monitor channel M^u_0
127	-	delayed monitor channel M^u_0

Table 5.11.: Memory layout of a pixels' context. Each context occupies 128 words of 512 words in a BRAM.

To efficiently use the multiplier and BRAM resources, the processing of a single block is done in a pipelined manner, which is shown in Table 5.10. The single multiplier processes all the 8 lags in each linear correlator block. With this scheme, 16 cycles are needed for the processing of a single block, referred to as a 'hardware-thread', and three further cycles for loading and storing the thread's context. According to the scheduling scheme (see section 2.4.2), a new input value may arrive once during the processing of two blocks or every 38 cycles. At a clock frequency of 144 MHz this means a new input value every 264 ns. All in all, each correlator is able to process data approximately 37 fold faster than the single pixel data rate (*i.e.*, frame rate) of the *RADHARD2* SPAD array. Therefore, sharing resources of a single correlator to process multiple pixels is possible, and each correlator is designed to compute the correlation function estimates of 32 pixels. Processing a single pixel is referred to as a 'hardware process'. Loading and storing the context of such a process takes five additional clock cycles, which is still within the available time-budget due to the high-speed of the correlator core.

Data structures

Table 5.11 gives an overview of the data structure of a single pixel's context. Each context comprises 128 data words, which are 48 bit wide. The lower 112 words are used to store the delay registers and the accumulators of 14 blocks with 8 lags each. The upper 16 words are used for data handover and monitor channels and cannot be used for correlation. Thus, only 14 linear blocks with 8 lags each are calculated.

Row reordering and context switching

To reduce the number of context switches, data acquired with the SPAD array had to be reordered to allow processing of a single row for a longer duration than just a single frame. The principle of this reordering is sketched in Figure 5.10:

The runtime of a hardware process (correlation of a single pixel) on the correlator is defined by the time required for the exchange of the currently unused contexts in the double buffering BRAM. In each context switch, the contexts of all 32 correlators have to be replaced by the next contexts. In total, $2 \times 32 \cdot 128 \cdot 48\,\text{bit} = 48\,\text{KiB}$ have to be transferred. To simplify internal logic, the second BRAM comprising 16 bit is treated as 32 bit, too. Thus, a total amount of 64 kB is transferred for each context switch. For a 32 bit wide static random access memory (SRAM) with an access time of 10 ns, this takes at least 164 µs. In that lapse of time, 621 raw input values can be processed (264 ns for each). Therefore, a context switch is done every 1024 input values (at least 270 µs).

In order to process 1024 consecutive values of a single row, all other rows have to be stored in the mean time. This is done on the first FPGA in 32 first in - first out buffers (FIFOs) of 32 bit width, which are implemented in the external SRAM. Under the assumption, that the correlator starts processing when all FIFOs are equally filled with 1024 values, the last row's FIFO has to be capable of holding additional $32 \cdot 1024 \cdot 264\,\text{ns}/10\,\text{ns} = 865$ values. Thus, a capacity of 2048 entries for every single FIFO, or 0.3 MB in total, is sufficient.

Additional implementation details

Data integrity. Data integrity during transport via the USB connection is ensured by using the same frame format as for the raw data , which includes a CRC-16 checksum. Each context that is transferred to the host computer is packed into a single frame. Therefore transmission errors can be detected and since the data is highly redundant, these errors can be corrected in most cases.

Optimizations. The correlator core operates at almost maximum clock frequency of the given FPGA hardware. To achieve timing closure, most signals are pre-calculated several cycles in advance and the final assignment is done with a latch. Furthermore, only a single state machine ('control', yellow, in Figure 5.11) is used for all 32 correlators.

Region of interest. The design described above can handle input data rates of up to 3.2×10^6 single lines per second. Either full frames can be processed with 10 µs integration time, or subregions down to two lines at higher speeds (625 ns integration time for a single line). Then only a reduced set of contexts is required. The selection of the lines in the region of interest (ROI) and the required reordering is done in the first FPGA.

Extended input stage. Although the correlator is limited to 10 µs integration time for full frames, the *RADHARD2* SPAD array can be operated down to 2.5 µs integration time. To take advantage of this higher frame rate, in between the SPAD array and the correlator an accumulation stage was introduced stage that sums up three consecutive frames. This way, the sensor is read every 3.3 µs and the probability for undetected double or triple photon events is reduced. Internally this is done by doubling the data path in the front end to 2 bit. The generic correlator itself operates on a 16 bit-wide data path and remained unchanged.

5.5.3. Simulation and testing

The entire correlator including SRAM was simulated using random datasets. After the correlation and the flushing, the contents of the memory was compared to the results of a software implementation of the correlator. In addition, the hardware correlator was tested with known input signals: Figure 5.12a shows a test measurement performed with the *RADHARD2* SPAD array and the FPGA correlator. A LED was used as light source, which was blinking with a fixed frequency of 2.5 kHz. The runtime of the correlator was 1.2 s. A fit of the autocorrelation curves with the oscillator model (blue, see section 2.4.3) yielded a frequency of (2507 ± 4) Hz. This along with the simulations positively showed the full functionality of the correlator hardware.

Further tests were performed within the microscope setup and known samples. Figure 5.12b shows a SPIM-FCS measurement of *TetraSpeck* $\varnothing = 100$ nm fluorescent beads beads. Frames were acquired at 3.3 µs resolution and temporarily binned, whereas the correlator ran at a temporal resolution of 10 µs. Although the detector is only suited for such bright samples, the diffusion of the beads could be determined accurately. Further details on this measurement as well as further measurements are shown in appendix E.4.

5.5.4. Resource utilization

Table 5.12 shows the resource utilization of the whole correlator. With 32 individual correlator cores for 1024 pixels, less than 50 % of the available resources are utilized. Although the final resource utiliza-

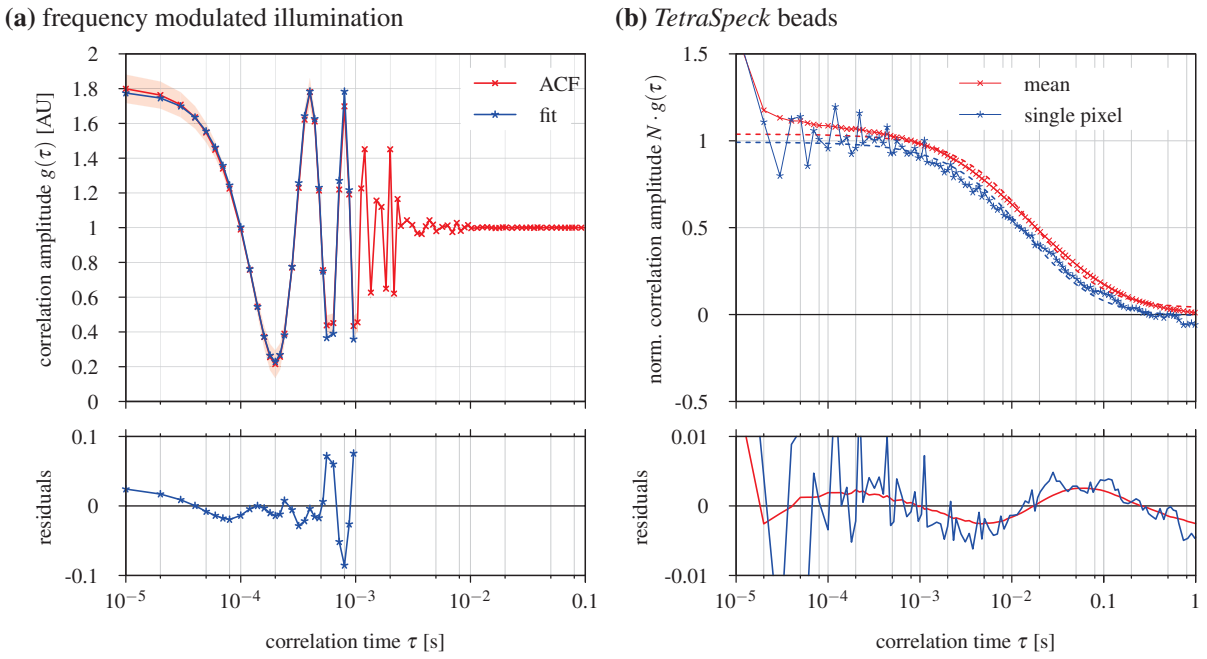


Figure 5.12.: Measurements performed with the hardware correlator. (a) Mean value of 1024 ACFs taken by the *RADHARD2* sensor, exposed to a 630 nm LED sine-modulated with a frequency of 2.5 kHz. The correlator was running for 1.2 s (131072 samples at $\tau_{\text{min}} = 10\,\mu\text{s}$). The theoretical model $g^{(\text{theoretical})}(\tau) = 1 + A \cdot \cos(2\pi f \cdot \tau)$ is fitted against the mean in the interval $[10\,\mu\text{s}, 0.6\,\text{ms}]$, and is shown until $\tau = 1\,\text{ms}$. The fit yields a frequency of $(2507 \pm 4)\,\text{Hz}$. The standard deviation of the 1024 curves is shown as light red area. (b) Autocorrelation curves of a measurement of *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads for a single pixel (blue) and the average over all pixels (red). See appendix E.4 for further details.

tion after the implementation of the design might have been higher*, one should be able to extend the design to 64 parallel correlators. This additional performance can either be used to improve temporal resolution, *i.e.*, by halving the number of pixels that are processed by a single correlator, or to calculate cross-correlation curves in parallel.

5.5.5. Conclusion

In total, the FPGA correlator achieved a performance of 1.6 Gflops/s. This is comparable to other implmentations based on the VIRTEX 2 FPGA, but an order of magnitude below the performance of commercially available correlators.

The correlator achieved 11 % of the peak performance of the FPGA and used less than half of the its resources. By doubling the number of correlators, one could further increase the performance. Assuming a linear scaling of the performance, such a correlator design should also achieve real-time performance for the CHSPAD, if a more recent VIRTEX-6 platform† is used. Under this assumption, the correlator would outperform any other hardware based correlator that has been proposed so far (*cf.*, section 3.2).

The correlator was successfully used in the SPIM setup (see section 2.3.1 for details on the microscope). Although the design is based on quite old hardware, the goal of real-time performance for the *RADHARD2* SPAD array was achieved.

A major disadvantage of an HDL-based correlator implementation was the tedious and time-consuming design and verification process. Additionally, the used hardware platform was quite limited regarding on board memory resources and interfaces.

The implementation does also not take advantage of the single photon input data. As for the first blocks multiplication of single bits is trivial, these blocks could be implemented as shift registers. That way, one could improve the temporal resolution or save DSP resources. Such a design using different implementations for different blocks was already described in Ref. [72].

5.5.6. Outlook

The implementation of the FPGA correlator is based on the 'naïve' algorithm (see section 5.2.1 for a description). As shown for the CPU and GPU platform, the streaming algorithm achieved a higher performance on parallel architectures if sufficient memory is available. But due to the limitations of the hardware platform, this approach could not be implemented. In the following section a correlator design based on the streaming algorithm is proposed which follows the dataflow paradigm.

5.5.7. Dataflow computing

category	occupied ressources	utilization [%]
registers	14×10^3	36
LUT	16×10^3	41
multiplier	32	16
BRAM	66	34

Table 5.12.: Resource consumption of the FPGA correlator (entity name 'correlation_processing_unit'). Further logic used for interfacing, *etc.*, is not considered. Data according to SYNPLIFY'S output after synthesis. Values differ slightly after design implementation with XILINX tools due to low level optimizations. Utilization is relative to the resources of the FPGA.

*If free resources are available during the implementation phase, those are typically used to duplicate logic. This can improve performance as data paths can be shortened.

†Assuming a ratio of 20 % of the theoretical peak performance and a VIRTEX 6 *XC6VSX475T* FPGA ($\sim$ 240 Gflops/s).

The paradigm of dataflow computing, that was first mentioned in the 1960s, models a program as a directed graph. Data then flows along the connections between various nodes, which execute dedicated operations on the data stream (MISD). All operations are run in parallel if input data is available [206]. Figure 5.13 shows an example for modeling the calculation of the mean value: First, input data q_i is forwarded to two nodes. One of the nodes counts the number of input values, the other node reduces the input data stream to the sum of values obtained so far. Finally, a third node determines the quotient of its two input values and passes the result to the output μ

Several programming languages allow for modeling in the dataflow paradigm [104]. In this context, FPGAs are quite interesting, as the dataflow paradigm maps well onto their inner structure with a huge amount of parallel resources. As shown in the next section, there are also hardware and software tools available that allow for direct programming reconfigurable hardware in the dataflow paradigm without the need of an hardware description language (HDL). If an algorithm can be described as a dataflow graph, this could be a promising method to create an FPGA design in a much shorter time than required for a manual design using an HDL.

MAXELER TECHNOLOGIES *MaxWorkstation*

An example of a software and hardware ecosystem especially designed for dataflow computing are devices sold by MAXELER TECHNOLOGIES [3]. Their FPGA-based accelerator cards, called dataflow engines (DFEs), are based on XILINX VIRTEX-6 FPGAs (*e.g.*, *XC6VSX475T* on the *MAX3* board), that are connected to the host computer via PCIe with a bandwidth of 2 GB/s. The FPGA has access to 24 GiB RAM with a bandwidth of 38.4 GiB/s [37]. Multiple of these DFEs can be interconnected via a proprietary link (*MaxRing*).

Software development is done with *MaxJ*, a *Java* dialect, specifically designed for programming in the dataflow paradigm. The *Java* code is compiled into VHDL by the *MaxCompiler* which is then synthesized with the XILINX design tools and finally executed on the FPGA cards.

Modeling the multiple-τ correlation algorithm as a dataflow graph

At a first glance, the multiple-τ correlation algorithm is already described as a directed graph (*cf.*, Figure 2.27): In a single linear correlator block, intensity values flow from the input through several lags towards the output. The entire correlator is made up of a chain of such linear blocks.

A major difference is that the output of the correlation does not happen at the end of the graph but is acquired perpendicular to the direction of flow (*cf.*, Figure 5.14). Furthermore the output of the last block of the correlator is discarded. Also data between two adjacent blocks is not handed over in every step but only in every second, which was the starting point for the optimizations used on other platforms.

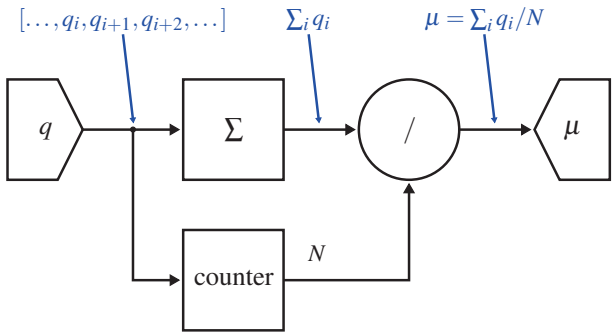


Figure 5.13.: Dataflow graph for a mean-value calculation. The directed graph comprises an input q, a counter, a stream operator /, a accumulating element $\sum$, and an output μ. It describes the calculation of $\mu = \sum_i q_i / N$ for all supplied input values q_i. The values of N are generated by the counter. Adapted from Ref. [76].

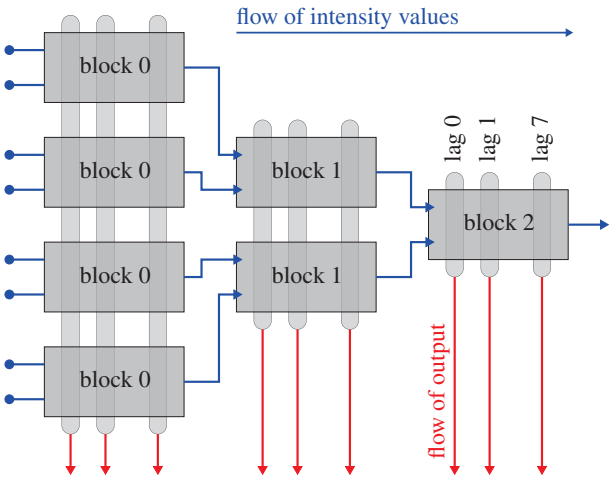


Figure 5.14.: Schematic drawing of a "sub-correlator" used for the dataflow implementation of the multiple-τ correlator. The shown "sub-correlator" comprises three linear correlator blocks. In each time-step, eight consecutive input values (pairs of local and global values each) are processed and a single value pair is provided as output. The result of the correlation are the states of the accumulators (red) that are retrieved perpendicular to the direction of flow of the intensity values (blue).

The concept of multiple pixels that share the same hardware correlator, as it was used for the FPGA implementation (see section 5.5.2), is not compatible with the dataflow paradigm and requires state machines and context switches (*i.e.*, an alteration of the graph and the states of its nodes during operation). But even if these problems are solved, the performance limitations of the FPGA remain: Each lag of the correlator consists of a multiplier and an adder circuit, which would be a natural choice for a single dataflow core. Then, the total number of cores is limited by the number of multipliers that are available in the FPGA (not considering the expensive implementation of such a function in the logic fabric). Even if all these could be saturated in parallel, the theoretical maximum performance (*cf.*, section 5.5.2) would only be twice the real performance of the GPU correlator (see Table 5.6). This is achieved only if the hardware can be run at the maximum clock frequency allowed by the FPGA. According to MAXELER TECHNOLOGIES, "dataflow computing achieves high performance through employing massive parallelism at low clock frequencies" [147], so the final speed of the design might be much slower (of the order of 200 MHz, *i.e.*, Ref. [77]).

Implementation

A possible implementation of a dataflow-optimized correlator for the MAXELER TECHNOLOGIES *MAX3* is as follows: Instead of reducing the execution rate by a factor of two between subsequent blocks, the execution rate of the previous block can be doubled. As the base clock for the hardware is constant, this can be achieved by doubling the hardware. One possible solution is the vertical unrolling described in section 5.2.8. Here two (or more) consecutive input values are processed in parallel so that a subsequent block can be run at full speed. Multiple blocks can be series-connected in a binary tree-like structure, which is shown in Figure 5.14. A similar approach was used for the GPU-based correlator (*cf.*, section 5.4) as this implementation helps to reduce the memory bandwidth requirements.

With eight lags per block, 252 blocks can be implemented in hardware, which is sufficient for a correlator with 7 levels of linear blocks, if one DSP slice is used in every lag*. If a single correlator with 16 levels of blocks should be implemented, 131 072 single blocks are necessary. For a correlator with four levels of linear correlator blocks, 15 single blocks are needed. The available resources are then sufficient for 16 "sub-correlators", processing four blocks each (1920 DSP slices). To process the full

*XILINX VIRTEX-6 devices, *e.g.*, *XC6VSX475T* have 2016 DSP slices.

correlation, assuming a multiple-τ correlator with 16 linear blocks, data has to be transferred trough that structure four times. Only the last block needs to have the full data width of 16 bit. In the predecessors, the data width can be reduced by 1 bit between two blocks, so that the input is only 13 bit wide. If all 16 "sub-correlators" process the fourth cycle of a data stream,

$$(\underbrace{16 \cdot 13\,\text{bit}}_{\text{input}} + \underbrace{17\,\text{bit}}_{\text{output}}) \cdot \underbrace{16}_{\text{\#correlator-blocks}} \cdot \underbrace{2}_{\text{local and global value}} = 832\,\text{Byte} + 68\,\text{Byte} = 900\,\text{Byte} \tag{5.5}$$

of memory input/output is required in each cycle.

If the output is discarded and a 13 bit data path is assumed, running at a clock rate of 600 MHz, a memory bandwidth of 464 GiB/s is required (832 Byte per cycle). At a given limit of 38.4 GiB/s for memory input/output (IO), a peak performance of 95 Gflops/s is expected at a clock frequency of 50 MHz. This should be feasible in hardware, but would give a performance of 0.6× of the real time, which is slower than required.

The assumptions only apply for the last block (fourth block), which is executed less frequently. If 16 first order "sub-correlators" are processed in parallel and if the bandwidth is limited as described, one should achieve up to 941 Gflops/s. The final performance is expected to lie in between and should allow for real-time correlation.

Conclusion

Although the overall performance is limited by the achievable memory bandwidth, a dataflow implementation that allows real-time performance on MAXELER TECHNOLOGIES hardware (*i.e.*, XILINX VIRTEX-6 *XC6VSX475T*) should be feasible. Therefore, a correlator of 16 linear blocks is cut into four 'sub-correlators', that comprise four blocks each and are based on vertical unrolling. Again, one sub-correlator can be used to calculate all the required lags of a correlator, if appropriate scheduling of the data streams is applied.

The memory bandwidth of a sub-correlator depends on its position within the correlator. If multiple sub-correlators are combined into a single design and these are operating at different positions, a mean bandwidth should be achieved, that allows correlating the CHSPAD data in real time or faster.

In view of the high costs of the development of a dataflow correlator and given that the GPU-based implementation achieved above real-time performance, no efforts were made towards an implementation on the MAXELER TECHNOLOGIES hardware.

5.6. Summary and conclusion

Table 5.13 summarizes the performance of the multiple-τ correlators that were evaluated on different platforms. The GPU correlator outperformed all other implementations by far. It was approximately five-fold faster than the CPU correlator and 260× faster than the FPGA-based implementation. The GPU correlator also outdid all commercially available correlators by at least one order of magnitude as well as the correlators proposed so far. Moreover, the correlators described above are also one of the first that feature such a large amount of channels (*i.e.*, pixels).

For all implementation based on the streaming algorithm, which performed best, the memory bandwidth was the limiting factor. This limitation was partially circumvented by using vertical unrolling to calculate several time-steps at once. On the CPU up to six time steps could be combined to make full use of the existing AVX register-set. In case of the GPU two correlator blocks could be merged by utilizing vertical unrolling to avoid data transfer to GPU memory in between the two blocks. This idea was extended for the proposed implementation based on the dataflow paradigm: Up to four blocks are combined to efficiently use the available DSP slices.

platform	theoretical peak performance [Gflops/s] (FMA)	implementation	performance absolute [Gflops/s] (FMA)	performance rel. to peak-perf. [%]	speed-up over real time
CPU	217	'classic'	70	32	0.43×
		'streaming'	85	39	0.52×
FPGA VIRTEX-II	29	'classic'	1.6	6	0.01×
FPGA VIRTEX-6	1210	dataflow†	> 164	> 14	> 1.00×
GPU	2523	'initial'	229	9	1.40×
		'optimized'	419	17	2.55×
		'fully optimized'‡	557	22	3.40×

Table 5.13.: Comparison of different correlator implementations that were evaluated in the course of this thesis. The speed-up in comparison to the real-time requirements is shown in curled braces. In case of the CPU and the GPU values are given in FMA instructions, in case of the FPGA implementation, MAC operations are shown. Real time requirements are 164 Gflops/s. The streaming algorithm requires approximately 1 MB per frame or 117 GB/s for real-time correlation. The theoretical peak performance is a characteristic of the device/hardware. †: Suggested implementation. ‡: Assumption based on compute-kernel runtime only.

Memory bandwidth requirements were much lower on the FPGA platform, as the data path can be adapted to the required width. In case of the CPU or GPU, the implementation is limited to the generic data types. It proved to be advantageous to use the single precision floating-point values as these are better supported by current SIMD instruction set extensions. Assuming a correlator of 16 blocks and a 32 bit wide data path, more than half of the required bandwidth remains unused (the last block needs a 16 bit wide input).

6. Application of SPAD arrays for *in vitro* and *in vivo* measurements

In this chapter, the performance of the Swiss single photon avalanche diode array (CHSPAD) array with microlenses (*cf.*, section 2.1.4) in the SPIM-FCS setup is assessed (see section 2.3.1 for details on the microscope). Especially the interplay between the optics of the microscope, the microlenses, and the sensor, as well as the data analysis is examined. Comparable measurements were performed with the *RADHARD2* single photon avalanche diode (SPAD) array, too, these are shown in appendix E.4. Due to its limited detection efficiency and the lack of microlenses, which increased the photon detection probability (PDP) of the CHSPAD approximately ten-fold (see section 4.3.7), the *RADHARD2* was used successfully only for the measurement of bright samples.

The overall objective was the measurement of the spatially resolved diffusion of fluorescent molecules. This information can either be used to create parameter maps or to obtain statistical information from the sample. The method of choice was fluorescence correlation spectroscopy (FCS), therefore the fluorescence intensity signal was recorded at every single pixel. As a next step, the correlation curves were calculated for every pixel. By model fitting these curves, the diffusion times of the species were obtained. With that, and if the shape of the focal volume is known, the diffusion coefficient can be calculated, or vice versa. This can either be used to characterize an unknown sample, to characterize the focal volume, or to assess the effect of the microscope optics on the measured values.

6.1. Measurement protocol

In the following, the standard procedures for several aspects of a single diffusion measurement are described. If not stated differently, these protocols were followed. The steps are in chronological order and are based on the procedure described in Ref. [119].

6.1.1. Sample preparation

Sample preparation was done according to Ref. [119]. All used dyes (see section 2.2.4 for details) were measured in aqueous solution. Small bags made of fluorinated ethylene propylene (FEP) foil were filled with 30 µL to 50 µL of these dilutions and were then heat-sealed. Bags were then mounted from above into the water filled sample chamber of the instrument. HeLa cells were grown on fragments of glass cover slips and were hung from above into the sample chamber filled with cell medium.

6.1.2. Alignment

First, basic alignment of the optics was done as described in Ref. [119]. As a second step, the detection efficiency of the CHSPAD was optimized: To do this, the sensor was illuminated with a white LED through the microscope optics (*cf.*, Figure 2.20). The sample chamber was filled with water but with any sample removed. In conjunction with the microlenses, the microscope optics lead to an image from the CHSPAD that has a bright spot in it. By adjusting the pan and tilt of the sensor board this peak was centered (*cf.*, section 4.3.7).

Next, the overlap between the focal plane of the detection objective and the lightsheet was optimized by adjusting the gimbal mounted mirror (GMM). For this purpose, a dilution of *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$

fluorescent beads was used as a sample, as here single beads were visible in the live-view from the sensor. The lightsheet was then shifted until maximum contrast was achieved, which means a maximum overlap of the lightsheet and the focal plane of the detection objective.

Finally, to determine the molecular detection efficiency (MDE) parameters of the microscope, *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads embedded in a stiff gel were used to perform a bead-stack (*i.e.*, a z-scan of the gel) which was evaluated as described in section 2.3.4. The estimates of w_{xy} and w_z (if not used as free parameters in the model fit) were used for all FCS fits in the following.

6.1.3. Image acquisition

Image acquisition was done using the CHSPAD control software *ngsoft* (see section 4.2.3 for details), which was remote controlled by *QuickFit3* as main experiment control software (via the CHSPAD-TCP/IP camera plug-in, see section 2.3.3). To cover a wide area and to allow measuring for a long duration, an region of interest (ROI) of 512×4 pixels in the center of the sensor was chosen. With these settings, a single measurement contained 15.8×10^6 single frames that were captured during a period of 101 s.

6.1.4. Data treatment

Raw photon data, as output by *ngsoft*, was correlated using the 'correlator_vc_chspad_raw' program (based on the CPU implementation described in section 5.3). This program also performed a bleach correction of the raw data (see section 2.3.2 for details) and further created a statistical analysis of the raw data (total count rate, time-binned videos, *etc.*). The latter were used by *QuickFit3* for the evaluation process. For all samples except for the bright and non-bleaching 100 nm beads, the bleach-corrected correlation curves were used for analysis.

6.1.5. Data fitting

Correlation curves as obtained from the correlator were fitted using the models described in section 2.4.3. For the autocorrelation curves (ACCs), a three-dimensional normal diffusion model (equation (2.56)) was used and a two-focus FCS model (equation (2.62)) in case of the cross-correlation estimates. The afterpulsing was accounted for by adding equation (2.61) to the model function if required. The shape of the focal volume was taken from the bead-scan performed during the alignment step. Noise on the correlation curves was reduced, if required, by averaging neighboring correlation curves prior to the model fitting step (so called binning). Therefore, the mean correlation function estimate of a pixel and its direct neighbors in a region of 2×2 or 3×3 pixels was used for model fitting.

6.1.6. Data representation

Autocorrelation curves shown in this work are plotted normalized by the particle number N, *i.e.*, as $N \cdot g(\tau)$, where $g(\tau)$ are the correlation function estimates. The value of N was taken from the model fit. Cross-correlation curves, when shown, were scaled with the same factor as the ACCs. Mean diffusion coefficients were obtained as the mean value of a GAUSSIAN fit to a histogram of the diffusion coefficients of all pixels. Usually, the standard deviation from those fits is given as uncertainty or error of the mean value. If mean or median values of several measurements are given, the errors are also mean or median values of the standard deviation.

Diffusion coefficients were normalized to standard conditions ($\vartheta = 20\,^\circ\text{C}$, according to equation (E.1)).

Mean diffusion coefficient and relative standard deviation. Ideally, a homogeneous sample of a diffusing species has the same diffusion coefficient D in every evaluated pixel. Unfortunately, a certain noise is added to the intensity signal by various parts of the system. It is expected that all influences occur randomly and therefore the final parameters (usually N and D) are distributed normally. Resulting values are therefore mean value μ obtained from a GAUSSIAN fit of the parameter's histogram for all evaluated pixels. The fit additionally gives a measure of the spread of the values, the standard deviation σ, which is used in the following as error estimate.

To be able to compare the distributions of a parameter for different species, the relative standard deviation, also known as the coefficient of variation, was used. It is defined as:

$$c_\mathrm{v} = \frac{\sigma_\mu}{\mu}. \tag{6.1}$$

6.2. Settings and measurement conditions

For all measurements described in this chapter, the CHSPAD was operated at the same settings, if not stated differently. These are shown in Table 6.1. A ROI of 512×4 pixels was chosen as a trade-off that allows for relatively long acquisition times of 101 s but also covers a wide area of the sensor to get an insight in the homogeneity of it and the optical setup. The dimensions of the focal volume were estimated from bead-scans as described in the last section. For this analysis, the values for w_{xy} and w_z were obtained as mean values from four bead-scans. Defective pixels were excluded from all evaluations.

Typically, the laser was set to its maximum power ($P_\mathrm{laser} = 60\,\mathrm{mW}$) to get the best stability. The intensity was then reduced by an neutral density (ND) filter with OD = 0.5 in the light path. After the propagation through all optical components, the laser power at the center of the lightsheet was measured to be $260\,\mathrm{W/cm^2}$. This is roughly half of the light intensity of a confocal FCS microscope ($P_\mathrm{laser} = 5\,\mu\mathrm{W}$, $w_{xy} = 250\,\mathrm{nm} \rightarrow I_\mathrm{confocal} \sim 2500\,\mathrm{W/cm^2}$) and about a factor of two higher than typically used for SPIM-FCS with electron multiplying charge coupled device (EMCCD) cameras ($100 - 200\,\mathrm{W/cm^2}$) [119, p. 168].

Parameter	Value
V_OP	24.0 V
V_Q	1.0 V
V_Top	2.0 V
gating scheme	optimized, passive quenching
single measurement duration	15.8×10^6 frames (101 s)
ROI	512×4
w_{xy}	(450 ± 91) nm
w_z	(1348 ± 210) nm
effective laser power P_eff	19.2 mW (60 mW laser + 0.5 OD filter)
light intensity at the center of the lightsheet I_LS	$\sim 260\,\mathrm{W/cm^2}$

Table 6.1.: CHSPAD: Settings of various parameters used during data acquisition and data analysis. For details on the parameters of the CHSPAD see section 2.1.4.

6.3. Characterization measurements using fluorescent beads

This section describes several aspects of the evaluation of the diffusion of a sample at the example of *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads. Due to the high brightness of the sample, the *TetraSpeck* beads were also used for a characterization of the instrument. The insights gained in this section found a basis for the evaluations presented in the following sections. On the SPIM-FCS setup equipped with the CHSPAD, six measurements (so called 'runs') of the same sample were performed in a series with a ten minute interval in between, without changing the setup. All evaluations presented in the following were done with the same raw datasets.

For direct comparison, the same sample was also measured on a standard confocal FCS setup. An example of the obtained ACCs is shown in Figure 6.1a. From 21 consecutive 15 s measurements a diffusion coefficient of $D_{\mathrm{FCS,20W}} = (4.10 \pm 0.38)\,\mu\mathrm{m}^2/\mathrm{s}$ was obtained on the confocal setup. This is in good agreement with the reference value $D_{\mathrm{20W,th}} = (3.9 \pm 0.6)\,\mu\mathrm{m}^2/\mathrm{s}$ [154].

6.3.1. Evaluation of the autocorrelation curves

Figure 6.1b shows two representative ACCs of a single pixel, measured at different locations of the sensor. The red curve was measured in the significantly brighter center area of the CHSPAD, whereas the orange curve was measured at the edge of the sensor. Despite from a higher noise due to the approximately two fold reduced light intensity (and with that a higher afterpulsing component), the shift between the two curves indicated a significant change in the diffusion time.

Figure 6.2a shows the histogram of the diffusion coefficients obtained individually for each pixel by a fit of the ACCs. From GAUSSIAN fits of the histograms, a median diffusion coefficient for all six runs of $\bar{D}_{1\ldots6} = (2.5 \pm 1.0)\,\mu\mathrm{m}^2/\mathrm{s}$ was extracted. This value underestimates the expected diffusion coefficient of $(3.9 \pm 0.6)\,\mu\mathrm{m}^2/\mathrm{s}$ roughly by a factor of 1.5.

Contrary to the expectations for a homogeneous sample, the histogram in Figure 6.2a did not follow a GAUSSIAN shape. The relative standard deviation of $c_{\mathrm{v}D} \sim 30\,\%$ is more than twice off the theoretical value ($c_{\mathrm{v}} \sim 7\,\%$, see section 2.2.4). The source of this distortion becomes clear, when the dependence of D on the x-position is taken into account, which is depicted in Figure 6.2b. Although a homogeneous sample was used, D ranged from $1\,\mu\mathrm{m}^2/\mathrm{s}$ to $4\,\mu\mathrm{m}^2/\mathrm{s}$. Furthermore, a horizontal shift between the two measurements indicated a drift of either the sample or an optical component.

For the evaluations described so far, D and N were used as free parameters whereas the shape of the focal volume defined by w_{xy} and w_z was obtained from a bead-scan (*cf.*, section 2.3.4). Such a bead-scan is typically limited by the number of detected beads (in the order of thousand) and tends to lose precision towards the edges as fewer beads are detected in the outer areas (*i.e.*, reduced intensity of the CHSPAD). Although evaluations of these scans did not yield any significant inhomogeneity across the sensor area (see section 4.3.8), these parameters may not be as constant as expected. To get a more precise assessment, the focal volume can also be derived from a measurement of the diffusion of a known sample. If the diffusion coefficient is known, the particle number N and the lateral axis of the focal volume w_{xy} can be used as free parameters for a model fit of the ACCs. Figure 6.3 shows the x-dependence of the lateral half axis of the focus w_{xy} obtained by such a fit. The mean value of the evaluation described above was taken as the constant diffusion coefficient. The graph indicates a significant position dependence of w_{xy}, which was neither prominent in the bead-scans nor was detected with the EMCCD camera so far. An influence due to the shape of the lightsheet could be ruled out, as the ROI ran perpendicular to the direction of light propagation. A further evaluation of this inhomogeneity, which is shown in appendix E.1, suggested that the main cause of this effect was the overlap between the lightsheet and the focal plane of the detection objective under a small angle

Since it is known that the diffusion coefficient should be constant for a homogeneous sample, the variation of the measured D in the fits must originate in a variation of the focal volume (*i.e.*, of the

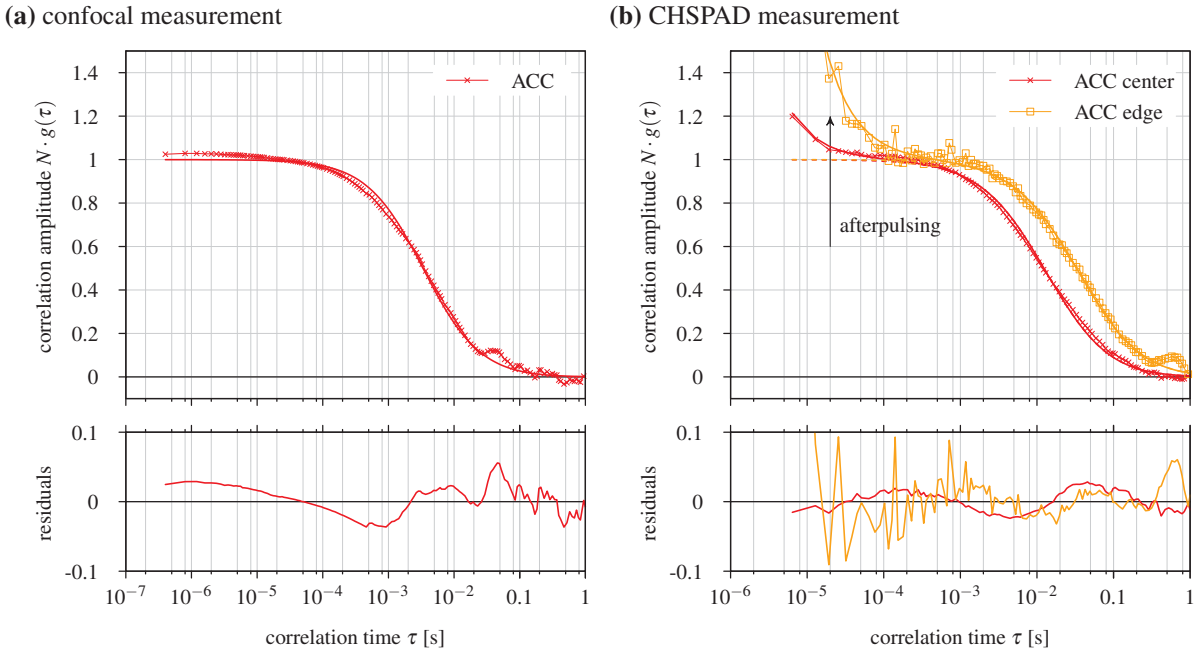


Figure 6.1.: Representative autocorrelation curves including fits and residuals of *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads measured on two different setups. (a) Confocal FCS setup using an *ALV 5000* correlator card. Measurement duration was 15 s. (b) measurement performed on the SPIM setup using the CHSPAD. The correlation curve is shown as a thin line, the corresponding fit as a solid curve. The model function without afterpulsing is shown as a dashed line.

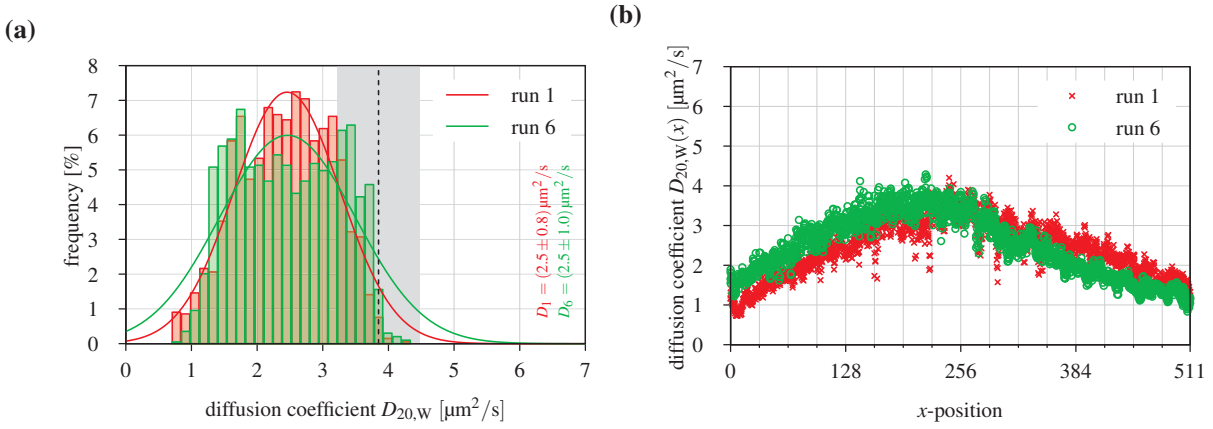


Figure 6.2.: Distribution of the measured diffusion coefficients D of *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads from fits of the autocorrelation curve. Two different measurements ('runs') of the same sample are shown in red and green. (a) Histogram of the diffusion coefficients D obtained for all pixels. Solid lines represent GAUSSIAN fits of the histograms. Fit results are given in the plot. (b) x-dependence of D. The theoretical value of $(3.85 \pm 0.63)\,\mu m^2/s$ [154] is indicated as a dashed black line, its error as a gray area.

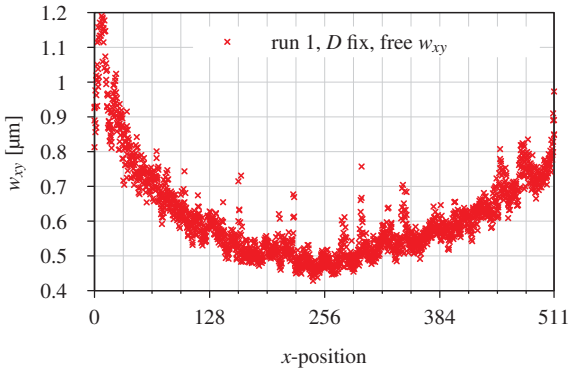


Figure 6.3.: Distribution of the lateral half width of the focus GAUSSIANw_{xy} of a *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads sample. Data is shown for run 1. The diffusion coefficient D was set to the value obtained from Figure 6.2.

parameters w_{xy} and w_z) across the sensor. Therefore, it would be beneficial to not only fit the parameters N and D, but also w_{xy}. If only the ACCs are used, this cannot be done, because D and w_{xy} are strongly correlated by the definition of the diffusion time:

$$\tau_D = \frac{{w_{xy}}^2}{4D}. \tag{6.2}$$

From the intensity traces obtained in a measurement, not only the autocorrelation curves can be calculated, but also cross-correlation curves with neighboring pixels. These are evaluated in the following section.

6.3.2. Evaluation of the cross-correlation curves

In addition to the autocorrelation curves, also the cross-correlation curves (CCCs) with neighboring pixels can be calculated by a correlator from the raw intensity traces. As the amplitude of the cross-correlation function (CCF) decays exponentially with the distance of the pixels, correlation curves of pixel with a distance above two pixels were not considered. In the following, the applicability for the evaluation of the diffusion of the CCCs with the left and right direct neighbor (one pixel distance, CCC 1) and the left and right neighbors with a distance of two pixels (CCC 2) is assessed*. As a fit model, the normal three-dimensional diffusion model was used for the ACCs (see section 2.4.3) and the two-focus FCS model for the CCCs (see section 2.4.3).

Figure 6.4 shows a set of correlation curves including fits for a representative pixel of the first run. As D and w_{xy} are no independent fit parameters (see above), the value for w_{xy} and w_z were kept constant for the fit of the ACC and the CCC 1. In case of the CCC 2, the spatial distance of the pixels reduces the correlation of D and w_{xy} and both were used free fit parameters. The plot shows, that the fit models correspond well with the correlation curves.

The histograms of the diffusion coefficients for two runs and both cross-correlation curves are displayed in Figure 6.5a and (b). The plots in (c) and (d) show the corresponding x-dependence of the diffusion coefficient. In comparison with the results of the fits of the ACCs (Figure 6.2), the distributions of the diffusion coefficient obtained by fits of the CCCs is much more narrow. The mean diffusion coefficient obtained from CCC 1 still underestimates the theoretical value by a factor of 1.5. If w_{xy} is used as a free parameter, the resulting diffusion coefficient obtained from CCC 2 is in range of the theoretical value. Figure 6.5e depicts the x-dependence of w_{xy} in the fits of the CCC 2. The shape of this curve is similar to the shape obtained from a fit of the ACCs with a constant D (see Figure 6.3). Its minimum is in accordance with the value obtained from the bead-scan.

*The raw data provided by the sensors is organized row-major, therefore only horizontal neighbors were taken into account.

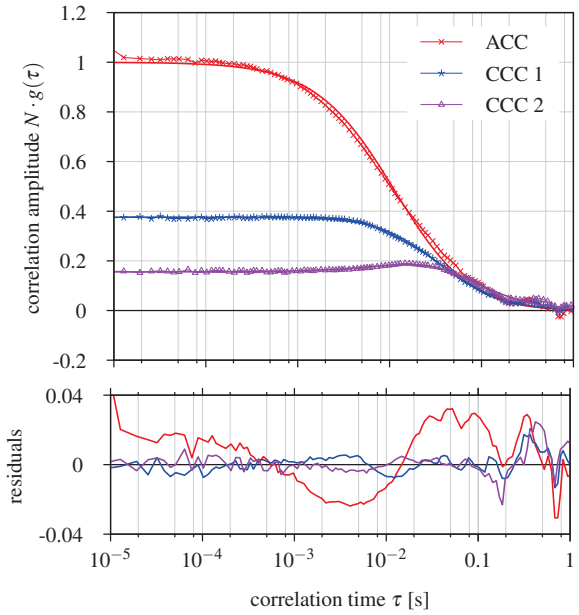


Figure 6.4.: Representative correlation curves including fits and residuals of *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads. The plot shows the ACC, CCC 1 and CCC 2 (thin lines) including individual fits (solid lines). For the fits of CCC 2, w_{xy} was used as a free parameter.

The narrow distribution of the measured D indicates that the fits of the CCC 1 are insensitive to variations of w_{xy}. On the contrary, if the focal volumes did not overlap, a fit of the CCC 2 allowed for the characterization of the lateral size of the focal volume and the diffusion coefficient simultaneously. If the diffusion coefficient is determined this way, the relative standard deviation slightly increases ($c_{\mathrm{vCCC\,2}} \sim 25\,\%$ instead of $c_{\mathrm{vCCC\,1}} \sim 20\,\%$).

To increase the precision of the results, all correlation curves discussed so far can be combined into a single global fit. Thus, all the advantages of every single fit of a correlation curve can be used. This method will be discussed in the following section.

6.3.3. Global fit of auto and cross-correlation curves

As shown above, model fits of single correlation curves were not sufficient to match the expected value of the diffusion coefficient. A major problem was the exact determination of the MDE and its inhomogeneity across the sensor area. To account for this, w_{xy} had to be used as a free fit parameter. In case of the CCC 2, this was already possible, but such fits of the ACC and the CCC 1 did not converge. Especially the ACC is important for a precise estimation of the particle number. To benefit from the advantages of every correlation curve, the model fitting of these were done simultaneously with the parameters N, D, and w_{xy} linked over the different model functions. Such a fit is called a 'global fit'. The well known separation between two neighboring pixels decouples w_{xy} and D in the CCFs (see equation (2.62)) and therefore all three parameters can be determined accurately in such a fit.

For the global fit, the minimization problem for a single correlation curve i shown in equation (2.64) turns into:

$$S_i(\beta_i) = \underset{\beta_i}{\operatorname{argmin}} \sum_{\delta_x \in \{-2,-1,0,1,2\}} \sum_{\tau} \left[G_i(\tau, \delta_{xi}, \beta_i) - g_i(\tau) \right]^2, \tag{6.3}$$

with the correlation curve g, the appropriate model function G, the horizontal pixel-pixel distance δ_x, and the parameter vector

$$\beta_i = (N_i, D_i, w_{xy_i}, w_{z_i}, \dots). \tag{6.4}$$

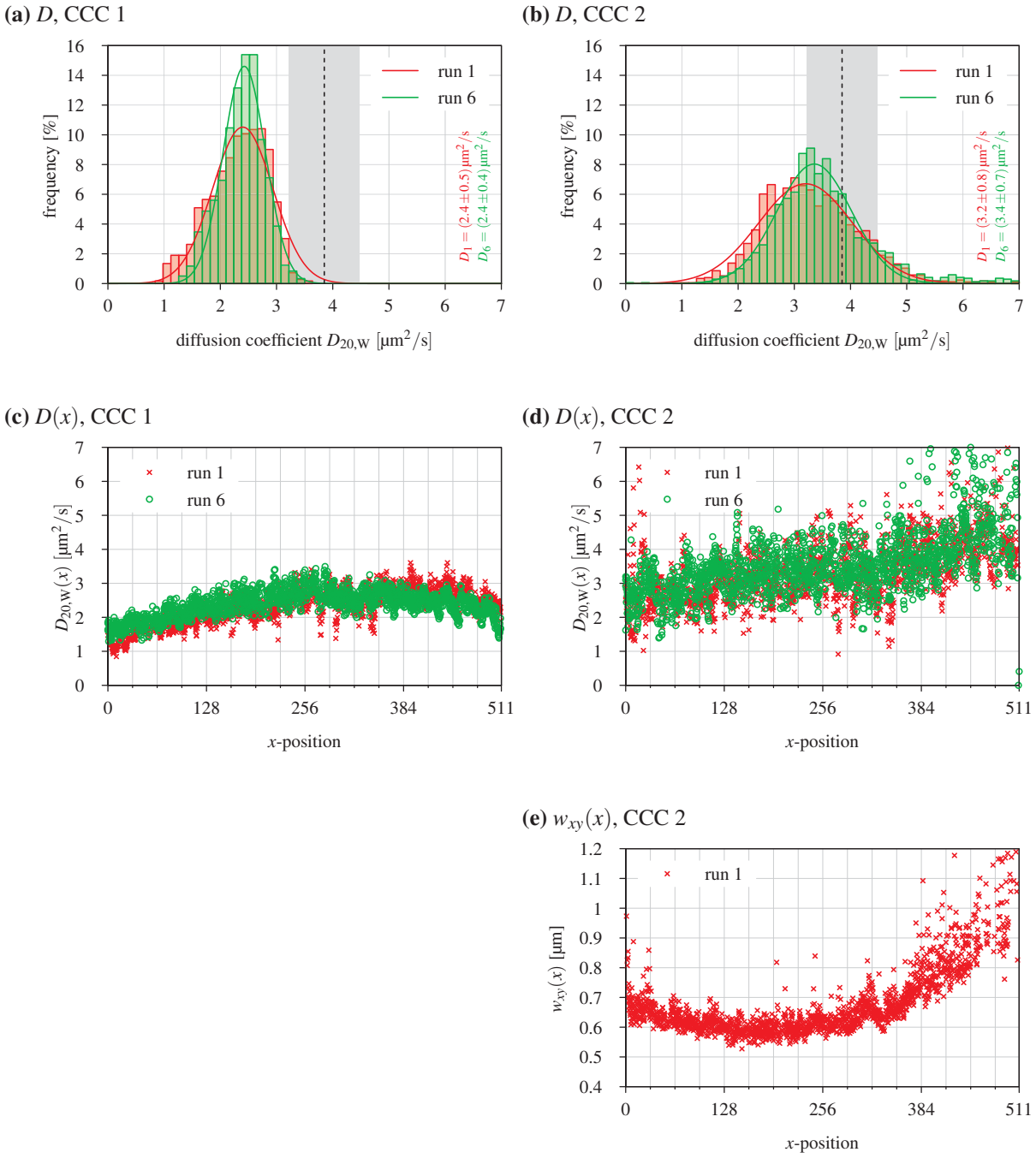


Figure 6.5.: Distribution of the measured diffusion coefficients D and the lateral half width of the focus GAUSSIANw_{xy} of *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads obtained from fits of the cross-correlation curves. Different measurements ('runs') of the same sample are plotted in red and green. (a) and (c) show evaluations based on CCC 1 with a constant w_{xy}, (b), (d) and (e) show evaluations based on CCC 2 with w_{xy} as a free fit parameter. (a) & (b) Histograms of the measured diffusion coefficients D obtained from fits of all pixels. Solid lines represent GAUSSIAN fits of the histograms. Fit results are given in the plot. The theoretical value of $(3.85 \pm 0.63)\,\mu\mathrm{m}^2/\mathrm{s}$ [154] is indicated as a dashed black line, its error as a gray area. (c) & (d) Dependence of D on the x-position of the CHSPAD. (e) x-position dependence of w_{xy}.

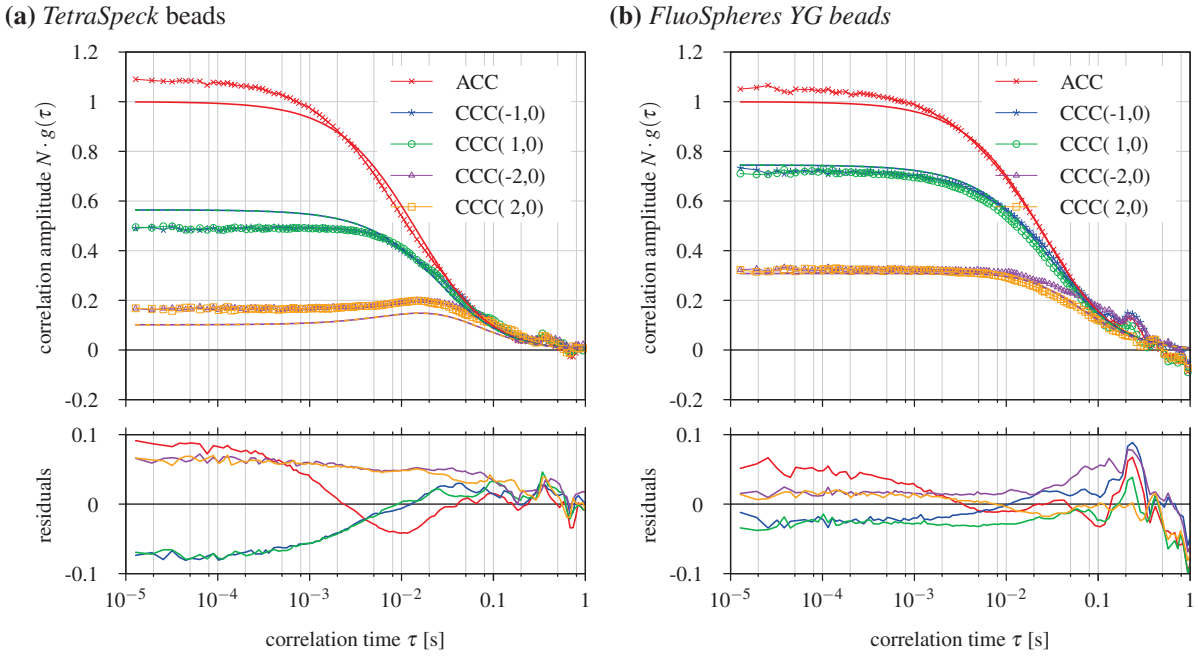


Figure 6.6.: Representative correlation curves of a single pixel for two different samples of fluorescent beads including global fits. The plot shows all five correlation curves including global fits and residuals. Thin lines represent the raw data, thick curves are the corresponding fit results. Results shown in (a) correspond to Figure 6.2.

If afterpulsing is accounted for, or an offset correction is done, further parameters may be added. The parameters of diffusion were assumed to be identical, thus the same particle number $N_i = N$, the same diffusion coefficient $D_i = D$, and the same focal volume ($w_{xy_i} = w_{xy}$ and $w_{zi} = w_z$) were used for all correlation curves in the global fit. During minimization, all correlation functions were treated equally and no weighting of the correlation curves was done. It should be noted, that a global fit that takes CCCs into account is only valid under the assumption that the parameters and diffusion properties are constant for all involved pixels. Otherwise, this method introduces a certain averaging.

To account for the inhomogeneity of the MDE, it is in principle possible to calibrate the focal volumes at each pixel with a known sample. But as indicated above, the variation seems not to be constant for multiple measurements, which can be seen as a drift in Figure 6.2b. Furthermore, changing the sample also influences the optical system, as the refractive indices of the different samples may not be identical. Also the detection efficiency of the CHSPAD was limited, so that a method not based on calibration was preferred.

Figure 6.6 shows the result of a global fit of two representative pixels for two different samples of fluorescent beads. The curves shown in (a) correspond to the measurement presented in Figure 6.1. Figure 6.6b shows a measurement of *FluoSpheres YG* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads. The fits are in good agreement with the correlation curves. A slight over- and underestimation might stem from an inaccurate estimation of w_z (obtained from bead-scan).

Figure 6.7 shows the histograms of the diffusion coefficients obtained by fits of different sets of correlation curves. The results of fits of the ACCs are shown in red. The histograms in blue and green were obtained from fits of the CCCs. For CCC 1 (blue), again only D and N were free fit parameters and w_{xy} was still taken from the bead-scan. For this short distance, the two focal volumes still overlap and D and w_{xy} are still correlated. This is relaxed only for the two-pixel distance in CCC 2 (green). Here N, D and w_{xy} could be used as free fit parameters. Finally, the histogram of the diffusion coefficients obtained by a full global fit to the autocorrelation curve and the left and the right cross-correlation curves with one and

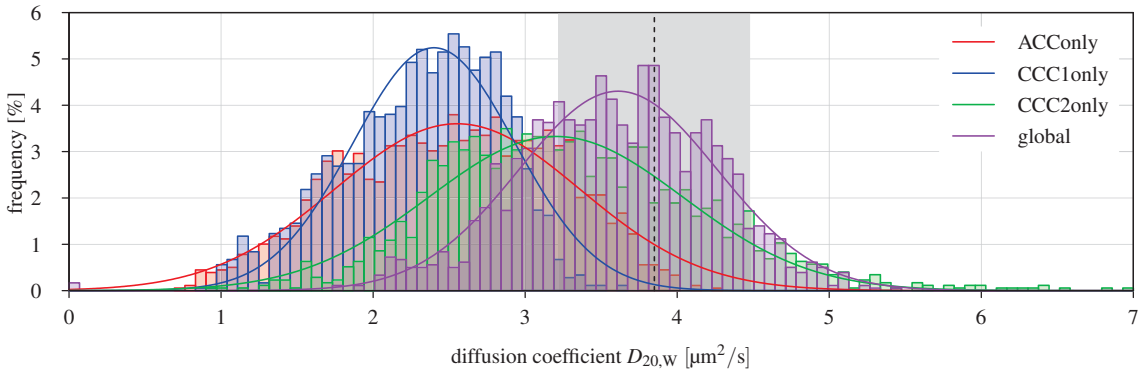


Figure 6.7.: Histograms of the measured diffusion coefficients D of *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads obtained from fits of different sets of correlation curves. A GAUSSIAN fit (solid curve) is applied to each histogram (not all are shown). The suffix 'only' denotes a model fit of a single correlation curve. 'ACC +CCC 1' denotes a global fit of the ACC and the left and right CCC. In a global fit, w_{xy} was used as a free parameter. In case of cross-correlation estimates, both, left and right neighbors were taken into account. The curves of CCC 2 and CCC2only did not decay to zero, therefore an offset-correction was added to the model function. For the CCC2only fits, w_{xy} was used as free parameter. Data is shown for run 6 (*cf.*, Figure 6.1b and Figure 6.2). Result from global fit: $D_{\mathrm{global}} = (3.6 \pm 0.7)\,\mu\mathrm{m}^2/\mathrm{s}$. The theoretical value of $(3.85 \pm 0.63)\,\mu\mathrm{m}^2/\mathrm{s}$ [154] is indicated as a dashed black line, its error as a gray area. For further evaluations of different samples see Figure E.3.

two pixel distance is shown in magenta. A solid curve represents the GAUSSIAN fit of each histogram. Further evaluations of different samples are shown in appendix E.2.

Figure 6.7 shows that by fitting a single correlation curve only (red, blue and green in Figure 6.7), the resulting diffusion coefficients systematically underestimated the expected value (dashed black line). This can be explained by the fact that a bead-scan may not be sufficient to assess the parameters of the focal volume with good accuracy. In case of the ACCs, the inhomogeneity in w_{xy} lead to a broadening of the histogram of the diffusion coefficients. On the contrary, the CCC 1, seems to be mostly insensitive to this inhomogeneity and showed a significantly narrower GAUSSIAN distribution. But still its mean value underestimated the diffusion coefficient. The histogram of the measured diffusion coefficients that were obtained by a fit of the CCC 2 tended to be broader than those obtained by a fit of the CCC 1, but were almost in range of the expected value.

By optimizing all correlation curves simultaneously in a global fit, the resulting distributions were in good agreement with the expected value (see Table 6.2 for a collection of the values). As shown in the last section, the fit of the CCC 2 (in contrast to the ACC) is more sensitive to the lateral size of the focal volume w_{xy}. On the contrary, a model fit of the ACC allows for a more precise estimation of N and D if the parameters of the focal volume are known with a good accuracy. Finally, the combination of all five correlation curves yielded the best results (*cf.*, Figure 6.7).

Going back to the example mentioned above, Figure 6.8 shows the result of global fits for the measurements already presented in Figure 6.1. The distribution of measured diffusion coefficients (Figure 6.8a) was much more narrow with an relative standard deviation below $c_{\mathrm{v}D} = 20\%$, which is significantly lower tan the results shown in the previous section ($c_{\mathrm{v}D} > 30\,\%$). In the horizontal distribution of diffusion coefficients (b) still a position dependence is prominent, but this dependence is way weaker than before and D only varies between $2.5\,\mu\mathrm{m}^2/\mathrm{s}$ to $5.0\,\mu\mathrm{m}^2/\mathrm{s}$.

The x-dependence of the fit-parameter w_{xy} is shown in Figure 6.8c. For comparison, the orange curve in Figure 6.8c was obtained from a fit of the ACC only where a constant diffusion coefficient (from global fit) was assumed and w_{xy} was used as a free parameter. Both showed a similar shape. The minimum of both curves was in agreement with the result of the bead-scan. The derived concentration in the focus c, is plotted in Figure 6.8d. Contrary to the expectations, the measured c is not constant for the homogeneous

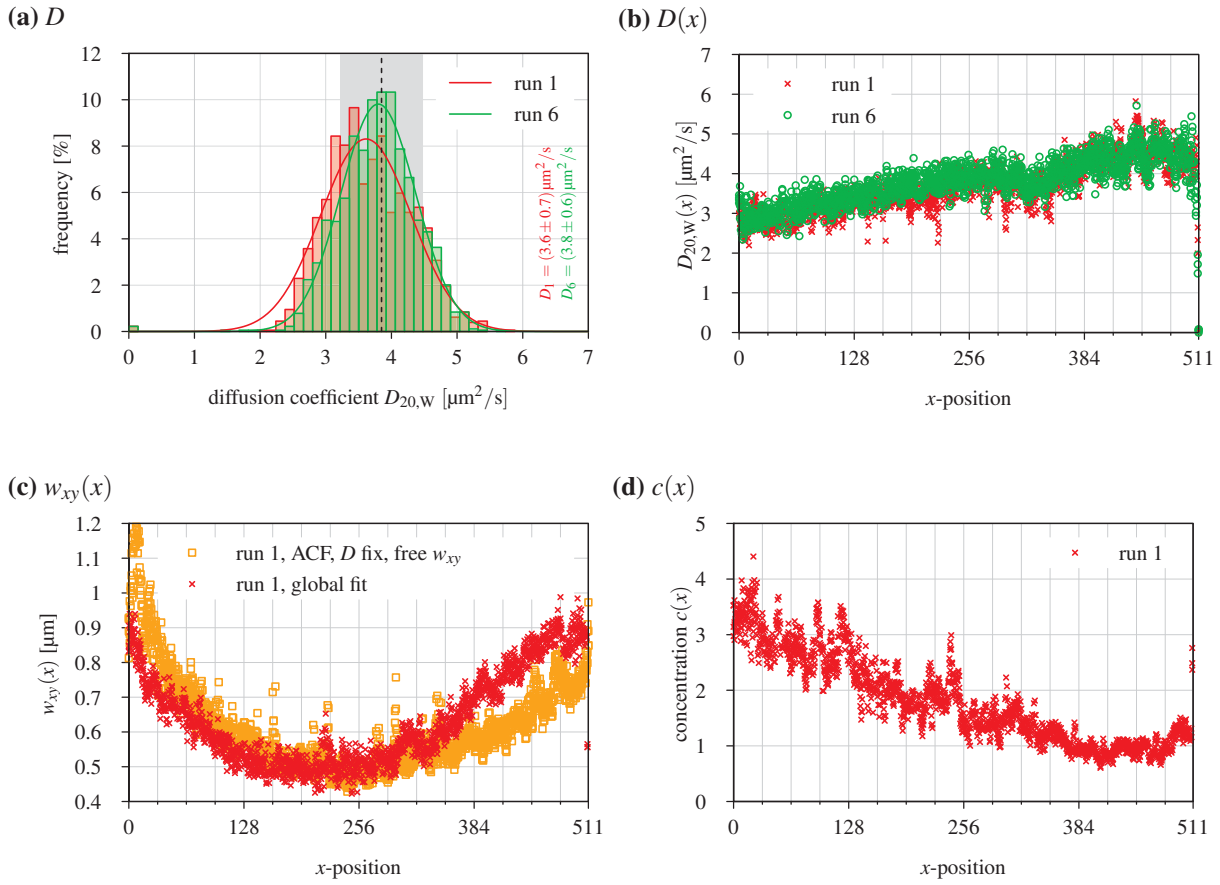


Figure 6.8.: Distribution of the measured diffusion coefficients D, the lateral half width of the focus GAUSSIANw_{xy}, and the particle concentration c of *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads obtained from global fits. Different measurements ('runs') of the same sample are shown in red and green. (a) Histogram of the diffusion coefficients D obtained for all pixels. Solid lines represent GAUSSIAN fits of the histograms. Fit results are given in the plot. The theoretical value of $(3.85 \pm 0.63)\,\mu m^2/s$ [154] is indicated as a dashed black line, its error as a gray area. (b) Dependency of D on the x-position (horizontal). (c) Position dependency of w_{xy}. The data from Figure 6.3 are shown for comparison (orange). (c) Position dependency of the particle concentration in the focal volume.

sample. As c also depends on the longitudinal half axis of the focus,

$$c \propto N \cdot {w_{xy}}^{-2} \cdot {w_z}^{-1}, \tag{6.5}$$

the drift might stem from an inhomogeneity of w_z, which will be addressed in the next section.

6.3.4. Global fit with constant c

Until now, the concentration c in the fit models used for auto- and cross-correlation was a derived parameter. It is calculated from w_{xy}, w_z, and N (see section 2.4.3). In order to use w_z as a free parameter and c as a constant, the role of both is exchanged. According to the definition of the effective volume, equation (2.59), w_z can be written as

$$w_z = \frac{V_{\text{eff}}}{\pi^{3/2} \cdot {w_{xy}}^2}. \tag{6.6}$$

Whereby the effective volume can be expressed by the particle number and the concentration (*cf.*, equation (2.60)):

$$V_{\text{eff}} = \frac{N}{c}. \tag{6.7}$$

By substituting w_z with equation (6.6) and equation (6.7) in the fit models, modified models were derived that rely on D, N, w_{xy}, and c instead of D, N, w_{xy}, and w_z.

Figure 6.9 shows the result of a global fit with a constant concentration for the sample evaluated in the last section. Figure 6.9b illustrates the x-dependence of the diffusion coefficient. In comparison to the global fit with a constant w_z (see last section, Figure 6.8b), the curve is almost flat across the entire sensor. Thus, the corresponding histogram of the diffusion coefficients, as shown in (a), is much more narrow. A GAUSSIAN fit of the histogram yields a value for the relative standard deviation of about $c_{\text{v}D} = 10\%$. The mean diffusion coefficient $D_{1\ldots6} = (4.0 \pm 0.4)\ \mu\text{m}^2/\text{s}$ is in good agreement with the theoretical value.

Figure 6.9c and (d) show the horizontal dependence of both parameters of the focus. The progression of w_{xy} is almost identical to that of the global fit (see last section, Figure 6.8c). Contrary to the expectations, the global fit with constant c indicated a significant position dependence of w_z, which was not further investigated.

For this evaluation, the mean value of the concentration by a previously ran global fit of the same dataset (*cf.* Figure 6.8) was used as a constant. This value can also be determined by a different measurement technique (*e.g.* FCS, photometer, *etc.*). If the concentration is not known (w_{xy}, w_z, and c used as free parameters), global fits did not converge. Therefore, in the following evaluations the original models are used with a constant w_z determined by a bead-scan. This is justified as the influence of w_z is much less than that of w_{xy} in the model functions.

So far, the global fit with a constant concentration yielded the lowest relative standard deviation of the diffusion coefficient. In the following section, the theoretical minimum of the relative standard deviation is evaluated. Table 6.2 shows a comparison of the diffusion coefficients obtained by different fit approaches.

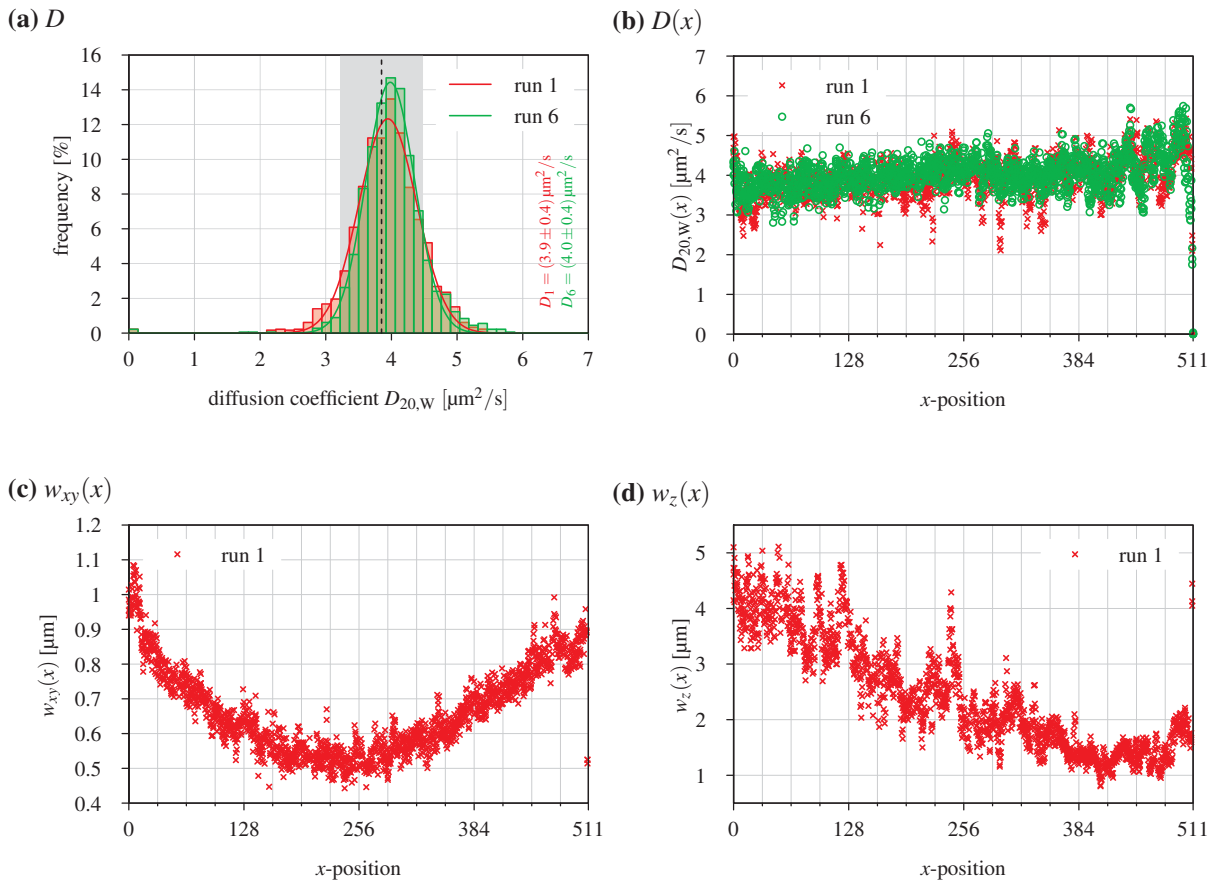


Figure 6.9.: Distribution of the measured diffusion coefficients D and the lateral and longitudinal half width of the focus GAUSSIANw_{xy} and w_z of *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads obtained from global fits with a constant concentration c. Different measurements ('runs') of the same sample are shown in red and green. (a) Histogram of the diffusion coefficients D obtained for all pixels. Solid lines represent GAUSSIAN fits of the histograms. Fit results are given in the plot. The theoretical value of $(3.85 \pm 0.63)\,\mu m^2/s$ [154] is indicated as a dashed black line, its error as a gray area. (b) x-dependence of D. (c) x-dependence of w_{xy}. (d) x-dependence of w_z.

6.3.5. Relative standard deviation of the measured diffusion coefficient

To get an impression of the theoretical relative standard deviation (rel. sd.) of the measurements shown above, the spread of the measured diffusion coefficients was evaluated in a local environment of the x-position. Therefore, the diffusion coefficient $D(x)$ is scaled at every position x with the local median of its ten pixel neighborhood:

$$D^{\text{detr.}}(x) = \frac{D(x)}{\text{median}\left(\{D(\chi), \chi \in [x-5, x+5]\}\right)}. \tag{6.8}$$

The resulting histogram is fitted with a GAUSSIAN function. For the fit parameters the following relations apply:

$$\mu_D \approxeq 1 \tag{6.9}$$

$$c_{\text{v}D} = \frac{\sigma_D}{\mu_D} \approxeq \sigma_D. \tag{6.10}$$

This so called 'de-trending' is used to estimate the spread of a signal that is not constant but slow changing in time or along an axis. As this method does not conserve the absolute value of D, only a statement about the relative errors can be made.

Figure 6.10a shows the distribution of the relative standard deviation of the measured diffusion coefficient after de-trending. The diffusion coefficients were obtained by fits of the ACCs only. The corresponding position dependence is shown in (b). For this *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads sample, the relative standard deviation of the diffusion coefficient was about 6 % on the average. A de-trending of the diffusion coefficients obtained by global fits yielded comparable results. Similar values of approximately $c_{\text{v}D} = 10\,\%$ were achieved for the fits of the ACCs in the center of the sensor without de-trending by choosing a 128 pixel wide subregion* (data not shown).

6.3.6. Comparison of different fit approaches

Table 6.2 shows an assembly of the measured diffusion coefficients based on fits of different sets of correlation curves. The values obtained by a global fit are in good agreement with the theoretical value

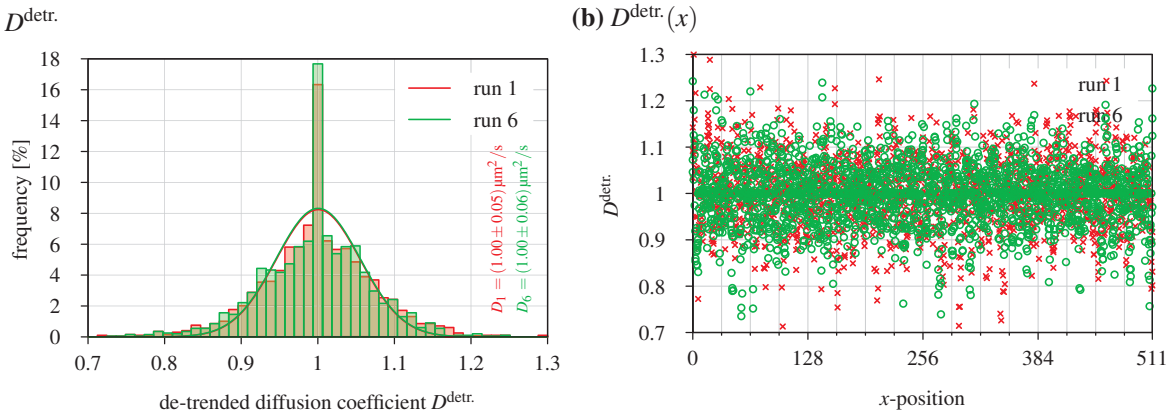


Figure 6.10.: Distribution of the de-trended diffusion coefficients $D^{\text{detr.}}$ of *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads obtained by fits of the autocorrelation curves. Different measurements ('runs') of the same sample are shown in red and green. (a) Histogram of the de-trended diffusion coefficients $D^{\text{detr.}}$ obtained for all pixels. Solid lines represent GAUSSIAN fits of the histograms. Fit results are given in the plot. (b) x-dependence of $D^{\text{detr.}}$.

*In principle, an arbitrary sized region can be chosen, but as the EMCCD camera has 128×128 pixels, this is a natural choice.

of $(3.85 \pm 0.63)\,\mu m^2/s$ [154]. A fit of the ACCs only underestimated the expected value by almost a factor of two.

By de-trending the measured diffusion coefficients along the lateral axis, a lower limit of 6 % for the relative standard deviation could be found. This is in agreement with the typical variations in the size of the beads ($c_v = 7\,\%$ [214]). The de-trended results are comparable for all different fit approaches.

The narrowest distribution of $c_{vD} \sim 10\,\%$ was reached by the global fit with a constant concentration. With a fixed w_z, the global fit yielded an relative standard deviation of less than $c_{vD} \sim 20\,\%$. Results that were obtained by fits of the ACCs only, had a relative standard deviation of 30% and above.

6.3.7. Conclusion

The evaluation of *TetraSpeck* $\varnothing = 100\,nm$ fluorescent beads revealed a possible position dependence of the focal volume. The exact cause of this inhomogeneity is not known to full extend. As indicated in appendix E.1, a possible source is the angle between the focal plane of the detection objective and the lightsheet. Other reasons might be optical aberrations that appear in the tube lens and the objective. These effects are visible only due to the large size of the sensor. The inhomogeneous light distribution across the CHSPAD due to the equidistant spacing of the microlenses or a divergence between the focal plane of the tube-lens and the focal plane of the microlenses seemed not to have a significant influence. As Figure E.2 shows, the same spacial variation of w_{xy} was visible for the sensors with and without microlenses. This effect was not detected so far with the four times smaller sensor of the EMCCD camera.

The results suggest, that by model fitting multiple correlation curves globally, it was possible to use w_{xy} as a free parameter and to account for the inhomogeneity in w_{xy} during the evaluation. It can be stated that the progression of w_{xy} along the x-axis is comparable to what can be obtained by a model fit of the ACC only with a fixed D and a free w_{xy}. At the same time the constancy of w_z has to be questioned, and it should in principle be used as a free parameter, too. Unfortunately, fits with such a set of free parameters did not converge.

Another solution would be to only use the inner 128×128 pixels of the sensor. A model fit of the ACCs only in this central area yielded a comparable (or even better) relative standard deviation than the global fit of the entire sensor. But at the same time this would eliminate the benefit of the huge size of the CHSPAD.

Finally, by using a de-trending of the measured diffusion coefficients, it could be shown that the setup is in principle capable of measuring diffusion coefficients of a homogeneous sample of fluorescent beads with a relative standard deviation of only 6 %. The remaining standard deviation might be caused by system-inherent noise or variations of the sample itself.

fit	run 1			run 6			median		
	$D \pm \sigma_D$ [$\mu m^2/s$]	c_{vD} [%]	$c_{vD}^{detr.}$ [%]	$D \pm \sigma_D$ [$\mu m^2/s$]	c_{vD} [%]	$c_{vD}^{detr.}$ [%]	$D \pm \sigma_D$ [$\mu m^2/s$]	c_{vD} [%]	$c_{vD}^{detr.}$ [%]
ACC (w_{xy}, w_z fix)	2.5 ± 0.8	33	5	2.5 ± 1.0	42	6	2.5 ± 1.0	39	5
CCC 1 (w_z fix)	2.4 ± 0.5	22	-	2.4 ± 0.4	15	-	2.4 ± 0.4	19	-
CCC 2 (w_z fix)	3.2 ± 0.8	26	-	3.4 ± 0.7	20	-	3.3 ± 0.8	24	-
global (w_z fix)	3.6 ± 0.7	19	5	3.8 ± 0.6	15	5	3.7 ± 0.6	16	5
global (c fix)	3.9 ± 0.4	11	6	4.0 ± 0.4	9	6	4.0 ± 0.4	11	6

Table 6.2.: Measured diffusion coefficients D of *TetraSpeck* $\varnothing = 100\,nm$ fluorescent beads obtained by fits of different set of correlation curves. The first column of every set represent the mean value of a GAUSSIAN fit of the histogram of the diffusion coefficients. Its relative standard deviation is shown in the second column. The third column of every set gives the relative standard deviation of the de-trended diffusion coefficient ('detr.'). The median value accounts for all six runs.

6.4. Influence of the detected light intensity and the binning on the measured diffusion coefficient and the particle concentration in the focus

A major problem of the detector is its limited photon detection efficiency. When samples are measured, the remaining question is the quality of the extracted diffusion coefficients, and whether the emitted light is sufficient for a correct determination of the diffusion coefficient and its independence of binning by averaging correlation curves from neighboring pixels.

In Figure 6.11 these effects are simulated using *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads (*T7279*, INVITROGEN). The original dataset containing the raw photon information was artificially dimmed, by randomly removing photon events from the dataset. The raw data were acquired at mean laser intensity of the lightsheet of about $70\,\mathrm{W/cm^2}$. Figure 6.11a shows the dependency of the measured diffusion coefficient and its relative standard deviation on the light intensity of the lightsheet. The results suggest that binning does not have a big influence on the final value of the measured diffusion coefficient, but significantly reduces the width of their distribution, especially for low light intensities. By using 4×4 pixels for binning, the same standard deviation as for the non-binned dataset can be reached, but already at an almost thirty-fold decreased laser power. Using a region of 2×2 pixels, a ten-fold lower intensity resembled this relative standard deviation. The plot also indicates that there is a lower limit of the relative standard deviation at about $c_{\mathrm{v}D} = 15\,\%$ which cannot be decreased any further by a higher light intensity. This minimum might be inherent to the sample, to the entire detection system, or to the fit procedure (compare last section). For this type of sample, a threshold of $c_{\mathrm{v}D} = 20\,\%$ relative standard deviation can be defined which guaranteed an almost constant mean diffusion coefficient.

Figure 6.11b shows the measured particle concentration in the focus obtained from the same simulation. For c, the lower limit for the relative standard deviation seemed to be about 40 %. In comparison to the diffusion coefficient, the result of the measured particle concentration in the focus is less stable. Roughly below $10\,\mathrm{W/cm^2}$, the fits start to underestimate the value at high count rates. Binning can be used to act against this tendency. The effect of underestimating the diffusion coefficient for low light situations can also be seen in the measurements of *QDot-525 streptavidin ITK* and eGFP oligomers shown in the Appendix (see Figure E.5 and Figure E.9).

Figure 6.12 shows the relative standard deviation as a function of the count rate per molecule (CPM). The evaluation is based on the dataset already presented in Figure 6.11. The simulation showed, that for values $\mathrm{CPM} < 300\,\mathrm{Hz}$ the relative standard deviation of the measured diffusion coefficient increases. This value has been used as a measure for the accuracy of the obtained diffusion coefficient.

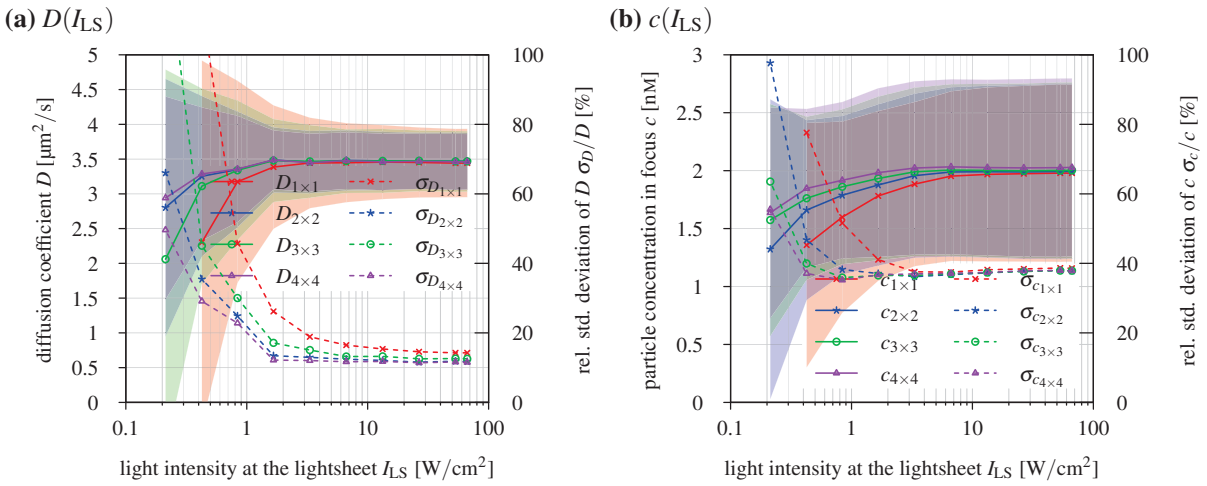


Figure 6.11.: Simulated influence of the detected light intensity on the measured diffusion coefficient and the particle concentration in focus of *TetraSpeck* ∅ = 100 nm fluorescent beads obtained by global fits. (a) Diffusion coefficient D. (b) Particle concentration in focus c. Results are shown for different bin-sizes (1×1 to 4×4). The respective relative standard deviation σ is shown as a dotted line, absolute standard deviation is shown as a semi-transparent area. All diffusion coefficients were re-calibrated to $\vartheta = 20\,^{\circ}\mathrm{C}$. A single measurement with an effective laser power $P_{\mathrm{eff}} = 5\,\mathrm{mW}$, which is approximately $I_{\mathrm{LS}} = 70\,\mathrm{W/cm^2}$ at the center of the lightsheet. Single photon events were then randomly removed from the raw dataset to stimulate measurements with a reduced excitation laser power.

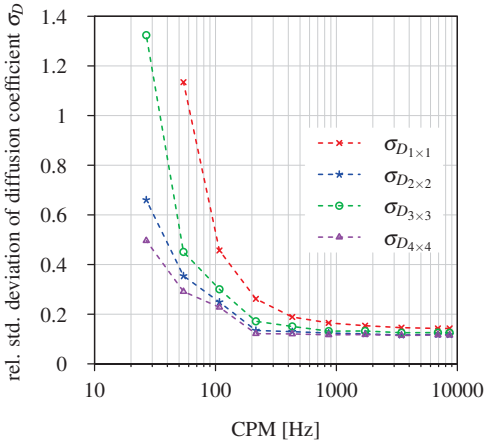


Figure 6.12.: Simulated influence of the count rate per molecule on the relative standard deviation of the measured diffusion coefficient of *TetraSpeck* ∅ = 100 nm fluorescent beads obtained by a global fit. Results are shown for different bin-sizes (1×1 to 4×4). Data is based on a single measurement with an effective laser power $P_{\mathrm{eff}} = 5\,\mathrm{mW}$, which is approximately $I_{\mathrm{LS}} = 70\,\mathrm{W/cm^2}$ at the center of the lightsheet. Single photon events were then randomly removed from the raw dataset to stimulate a reduced excitation laser power. Same data as also shown in Figure 6.11.

6.5. Dependency of the measured diffusion coefficient on the minimum lag time

SPAD arrays allow for much higher frame rates and therefore for shorter correlation times in FCS experiments. If the minimum correlation time τ_{min} is too high, the plateau of the correlation curve cannot be resolved and consequently the fits cannot accurately estimate the parameters anymore. The impact is highest for the particle number, which is proportional to $1/G(0)$. To be able to obtain reliable results, it is therefore important to know the minimum value τ_{min} which is necessary to resolve the parameters of interest with a reasonable precision.

Knowing the limits of τ_{min} is also valuable information for designing a readout system with respect to the maximum frame rate. But also for correlator design where every additional low-lag time correlator block increases the computational effort by a factor of two. If the single frame integration time is higher than the required τ_{min}, consecutive input values can be binned in the time domain and be correlated at reduced speed. In the following, the dependence of the measured diffusion coefficient D on τ_{min} is evaluated for two different samples: *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads and *QDot-525 streptavidin ITK*.

To simulate a slower detection system, all correlator channels with τ smaller than the desired τ_{min} were excluded from the dataset before the model fit. As the impact of the afterpulsing (*cf.*, section 2.4.3) is quite high, data is normalized to the measurement with the first five channels excluded (D_5, c_5). All datasets were fitted with the same models that include afterpulsing. Both samples diffuse quite differently, so only relative concentrations and diffusion coefficients were considered. The given errors are relative standard deviations of GAUSSIAN fits to the resulting histograms of the corresponding parameters obtained for each single pixels. A single measurement took 51.2 s (80×10^6 frames) with a field of view of 4×512 pixels. The first channel corresponded to $\tau_1 = 6.4\,\mu\text{s}$.

Figure 6.13 shows the measured relative diffusion coefficients and particle concentrations for the two samples in relation to the minimum τ_{min}/τ_D. These evaluations were done for fits of the ACCs only ((a) & (b)), and for the global fit ((c) & (d)). The standard deviation is shown as solid area, the relative standard deviation of the mean value is shown as a dotted curve.

When only the autocorrelation function (ACF) estimates were taken into account, the diffusion coefficients as well as the particle concentrations were less stable for an increasing τ_{min}/τ_D. For both samples, the relative standard deviation started to change significantly at roughly $\tau_{\text{min}}/\tau_D = 0.05$. For the *QDot-525 streptavidin ITK* sample, the width of the distribution of the diffusion coefficient is significantly better than for the *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads sample, which was opposite in case of the particle concentration. The same evaluations with the same samples were done using the global fit: When only the mean values of the distributions were considered, the diffusion coefficient was almost constant even for $\tau_{\text{min}}/\tau_D > 1$ in case of the *TetraSpeck* beads. For the *QDot-525* it started to change at $\tau_{\text{min}}/\tau_D = 0.1$.

Considering the concentrations, the mean value was more stable and could be determined for values above $\tau_{\text{min}}/\tau_D = 1$. In contrast to the fits of the ACCs, the measured concentration tends to underestimate the expected value for an increasing τ_{min}/τ_D. A similar study for the same sample using an EMCCD camera was published in Ref. [120].

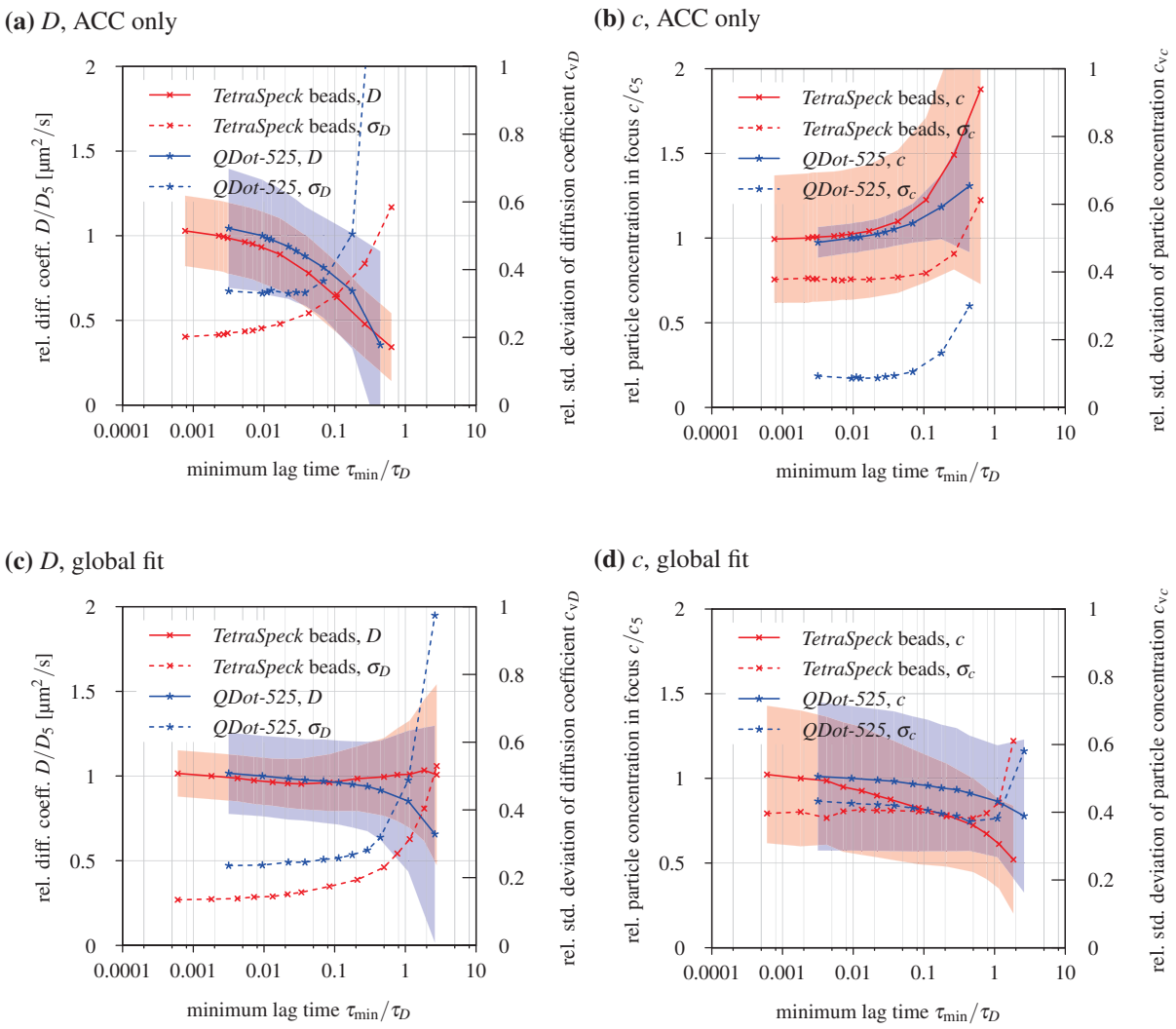


Figure 6.13.: Dependence of the measured diffusion coefficient D and the particle concentration c on the minimum lag-time used for recording. Plots (a) and (b) are based on fits of the ACCs, (c) and (d) on global fits. The effect is shown for two samples, *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads (red) and *QDot-525 streptavidin ITK* (blue). In both plots, the relative change of the fit parameter compared to the measurement without the first five channels is shown in relation to τ_{min}.

6.6. Measurement of the diffusion of small fluorescent dyes

The evaluations presented so far were done with relatively large and bright 100 nm fluorescent beads. In this section, smaller and thus faster dyes are evaluated in the SPIM-FCS setup for the use with the CHSPAD. These were the medium sized *QDot-525 streptavidin ITK* (*Q10041MP*, INVITROGEN) with a diameter of about 6 nm, *Alexa-488* (INVITROGEN), which is one of the smallest dyes (1 nm in diameter), and enhanced green fluorescent protein (eGFP) oligomers with diameters ranging from 4 nm to 12 nm (see section 2.2 for details on the dyes). Such fluorescent dyes are commonly used for the labeling of biological samples.

6.6.1. *QDot-525* and *Alexa-488*

Figure 6.14 shows representative examples of the correlation curves obtained for *QDot-525 streptavidin ITK* and *Alexa-488*, including global fits. In contrast to the significantly brighter 100 nm fluorescent bead samples, here the effect of afterpulsing became much more prominent in the ACCs. In case of the *QDot-525*, the afterpulsing was still well separated from the diffusion induced decay of the correlation curve. For the sample of *Alexa-488*, a fit of the ACCs did not yield any meaningful result as both decays were on the same timescale. The afterpulsing was not detected in the CCF estimates, as the read-out of different pixels is not correlated.

Figure 6.15 shows the corresponding histograms including GAUSSIAN fits of the measured diffusion coefficients for both samples. In both cases, reasonable narrow distributions were obtained for the non-binned data: $c_{\mathrm{v}D\mathit{QDot\text{-}525}} = 30\,\%$ and $c_{\mathrm{v}D\mathit{Alexa\text{-}488}} = 30\,\%$ for the mean values of three samples each. In comparison to the *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads (*T7279*, INVITROGEN) the distributions are about a factor of two wider ($c_{\mathrm{v}D\mathit{TetraSpeck}\ \mathrm{beads}} \approx 13\,\%$). This can be explained with the significantly lower brightness of both fluorophores and with that more noisy correlation curves.

The effect of afterpulsing could be corrected for using a global fit as already explained in the last section. Without a global fit, distinguishing between the afterpulsing and the diffusion was impossible in case of *Alexa-488*, as the decay of the afterpulsing is in the range of the diffusion time of the sample (see Figure 6.14c). Again, for both samples, additional binning significantly reduced the noise, consequently improved the fit accuracy, and almost halved the width of the distribution of diffusion coefficients (see Table 6.3 for a summary).

Additional measurements of the samples are shown in appendix E.3. The results of this measurement are compared to results of several other diffusion measurements in section 6.7.

6.6.2. eGFP oligomers

As a third sample eGFP oligomers were evaluated. These fluorescent proteins can be used to create fusion-proteins of a target protein and the dye which can then be integrated into live cells via transfection. The cell then starts to produce the fusion-proteins which can be investigated for example with fluorescence microscopy techniques. In this section eGFP oligomers were assessed in the SPIM-FCS setup. For this purpose, dilutions of the protein in aqueous solution were evaluated. Live cell measurements of eGFP oligomers are presented in the next section.

Figure 6.16 shows representative ACCs of two different eGFP oligomers with and without binning and results of a global fits (CCCs are omitted for a better readability). Again, as for *Alexa-488*, the diffusion time of eGFP monomers (red) overlapped with the decay time of the afterpulsing. For the four fold larger and brighter tetramers, both components could be separated easier. Due to the low fluorescence intensity, the curves without spatial binning showed heavy noise.

The corresponding histograms of the measured diffusion coefficients including GAUSSIAN fits are depicted in Figure 6.17. By applying spatial binning, the relative standard deviation was decreased by a factor of two down to $c_{\mathrm{v}D\mathrm{eGFP\text{-}1x}} \sim 30\,\%$ and $c_{\mathrm{v}D\mathrm{eGFP\text{-}4x}} \sim 16\,\%$. The mean values are in good

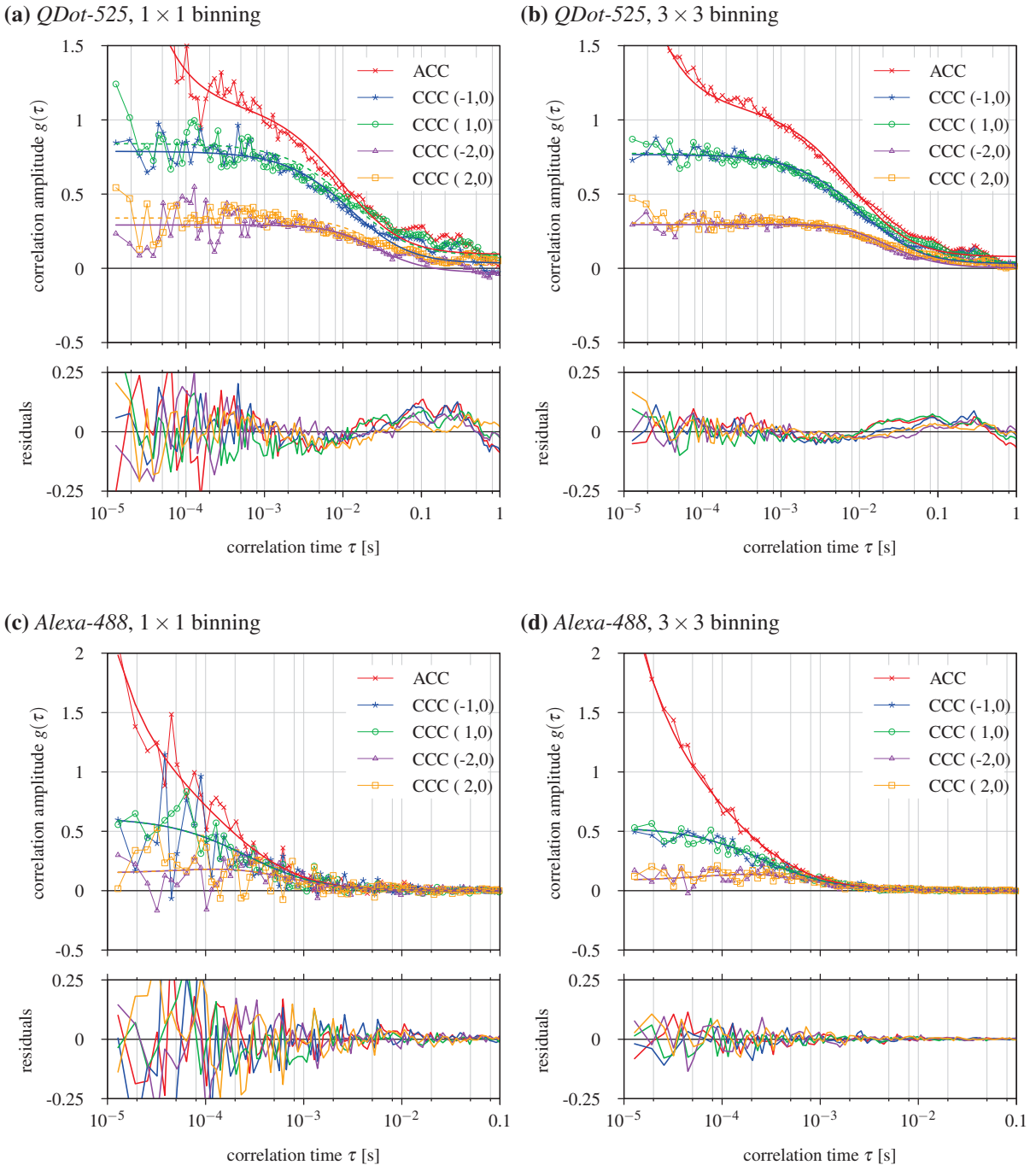


Figure 6.14.: Representative examples of single pixel measurements of *QDot-525 streptavidin ITK* and *Alexa-488* in solution with and without binning. The plots show all five correlation curves (*i.e.*, ACC + 4 CCCs) including global fits and residuals. Laser intensity at the center of the lightsheet was approximately $I_{LS} = 255\,\mathrm{W/cm^2}$. Thin lines represent the raw data, thick curves are the corresponding fit model.

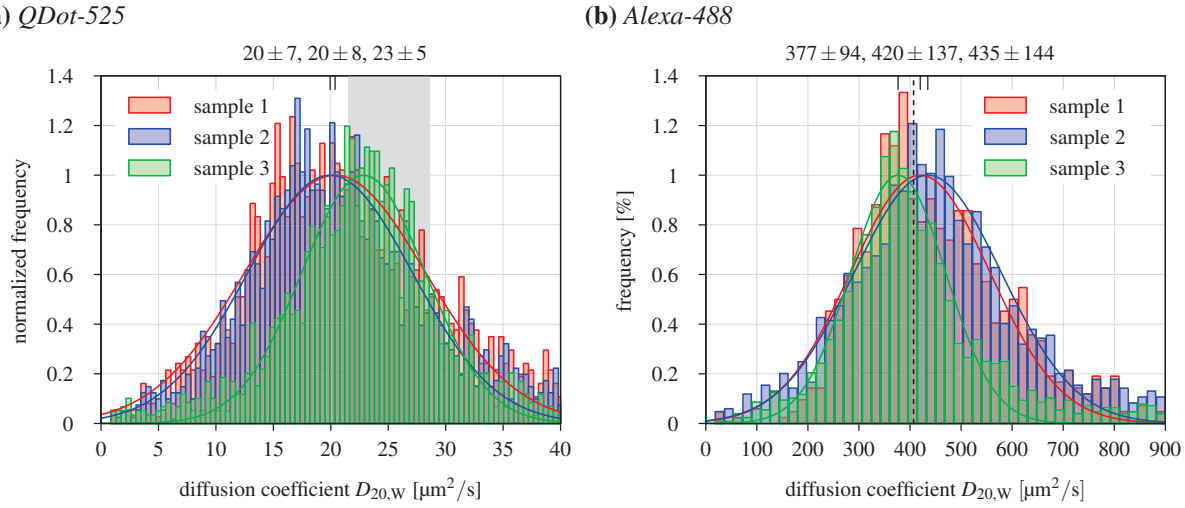


Figure 6.15.: Histogram of the measured diffusion coefficients D for three different samples of *QDot-525* and *Alexa-488* in solution. Diffusion coefficients were obtained by global fits. The mean values of the GAUSSIAN fits of the histogram are shown above the plot. (a) *QDot-525 streptavidin ITK*. Laser intensity in the center of the lightsheet was set to 80 W/cm² (red,blue) and 400 W/cm² (green). The gray rectangle in the background represents the expected diffusion coefficients of for spherical particles of a diameter 15 nm to 20 nm. No binning was applied. Mean value of the three measurements: $\bar{D}_{QDot\text{-}525} = (21 \pm 7)\,\mu m^2/s$. (b) *Alexa-488*. Laser intensity in the center of the lightsheet was set to 80 W/cm². Results are shown for three different dilutions. Mean value of the measurements: $\bar{D}_{Alexa\text{-}488} = (411 \pm 124)\,\mu m^2/s$. Raw data has been binned 3×3. The literature value $D_{20,W_{Alexa\text{-}488,\,lit.}} = 407\,\mu m^2/s$ [173] is shown as a black dotted line.

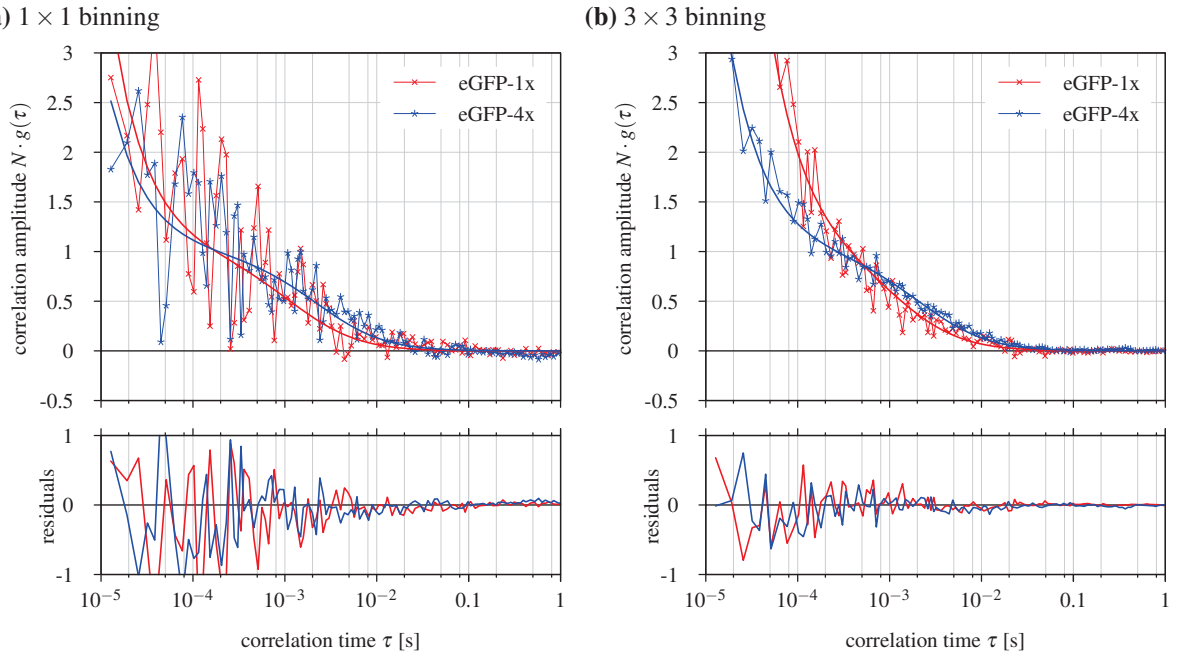


Figure 6.16.: eGFP oligomers in solution: Example of raw autocorrelation curves including global fits. Pixels were selected to show a diffusion coefficient similar to the mean value of each sample. Fits are shown as a thick line. The two plots show curves of the same pixels (a) without and (b) with binning.

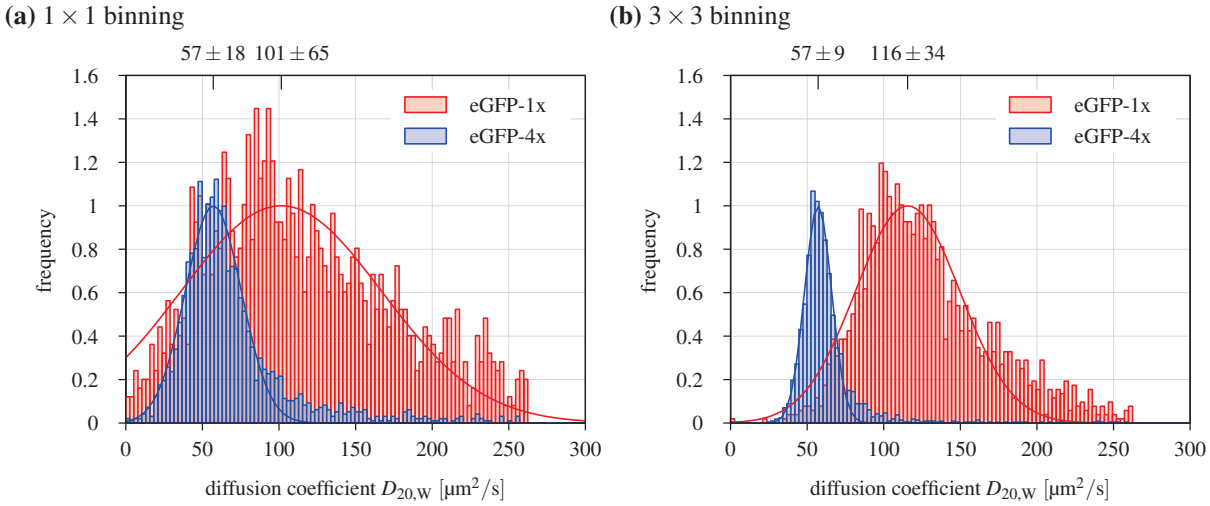


Figure 6.17.: eGFP oligomers in solution: Histograms of the measured diffusion coefficients obtained by global fits. The mean values of the GAUSSIAN fits (solid lines) of the histograms are written above the plot. Results are shown without (a) and out binning (b).

agreement with measurements of the same sample on the same instrument with the EMCCD camera [119]. Additional measurements of eGFP oligomers are shown in the appendix E.3. The results of the diffusion measurements of various samples are assembled in Table 6.3.

6.7. Summary of diffusion measurements

The diffusion measurements described so far, as well as further measurements described in appendix E, are summarized in Table 6.3. Only measurements of the fluorescent bead samples exceeded the threshold for the count rate per molecule of 300 Hz which is required for an accurate determination of the diffusion coefficient (*i.e.*, $c_{vD} < 20\,\%$, see section 6.4). All other samples had a significantly lower count rate per molecule, of the order of 100 or below. For all samples, 3×3 spatial binning reduced the relative standard deviation of the diffusion coefficient D approximately by a factor of two. Binning also lowered the risk of underestimating the diffusion coefficient of dim samples (*i.e.*, eGFP-1x, see also section 6.4). It can be stated, that all samples evaluated with the CHSPAD sensor yielded results that are in agreement with the references within the errors (see Table 6.3 for details on the reference values).

In case of the eGFP oligomers, the high background signal in relation to the fluorescence made it impossible to detect significant changes in the count rate per molecule for the different constructs. Especially for the dim monomers, afterpulsing had a significant influence on the count rate, so that the expected factor of three to four in the CPM between the monomer and the tetramer was not observable. However, the mean diffusion coefficients were nearly identical to the results that were obtained in a comparable measurement of the same oligomers on the instrument with the EMCCD camera [119].

sample	binning	$D_{20,\mathrm{W}}$ [µm^2/s]	$c_{\mathrm{v}D_{20,\mathrm{W}}}$ [%]	CPM [Hz]	$D_{\mathrm{theo.}}$ [µm^2/s]	comments
TetraSpeck beads	1×1	3.71 ± 0.45	12	~ 660	3.85 ± 0.63	1 measurement
FluoSpheres YG beads	1×1	4.09 ± 0.65	16	$\sim(3\ldots12)\cdot10^3$	3.85 ± 0.63	4 samples
	1×1	20.8 ± 8.4	40	$\sim 30\ldots170$	21 to 29	17 measurements
QDot-525	2×2	21.4 ± 6.3	29		21 to 29	18 measurements
	3×3	21.8 ± 5.2	24		21 to 29	18 measurements
	1×1	375 ± 119	32	$\sim 66\ldots127$	407	6 samples
Alexa-488	2×2	385 ± 79	21		407	6 samples
	3×3	391 ± 55	14		407	6 samples
eGFP-1x	1×1	102 ± 68	67	$\sim 34\ldots38$	118 ± 8	4 measurements
eGFP-1x	3×3	116 ± 34	29		118 ± 8	7 measurements
eGFP-2x	1×1	71 ± 43	61	$\sim 26\ldots37$	77 ± 9	4 measurements
eGFP-2x	3×3	77 ± 21	27		77 ± 9	4 measurements
eGFP-3x	1×1	59 ± 20	34	$\sim 32\ldots55$	N/A	4 measurements
eGFP-3x	3×3	61 ± 11	18		N/A	4 measurements
eGFP-4x	1×1	57 ± 18	32	$\sim 34\ldots61$	63 ± 6	4 measurements
eGFP-4x	3×3	57 ± 09	16		63 ± 6	4 measurements

Table 6.3.: Summary of measured diffusion coefficients of various samples performed with the CHSPAD. All diffusion coefficients were obtained from global fits. The given values for the diffusion coefficients are medians includin median values of the standard deviation (sd). Values for 100 nm beads are taken from Figure E.3. The theoretical value is taken from Ref. [154]. Detailed measurements of *FluoSpheres YG* $\varnothing = 100$ nm fluorescent beads are shown in Figure E.4. The theoretical value is assumed to be identical to the one of *TetraSpeck* beads. Detailed measurements of *QDot-525 streptavidin ITK* are shown in Figure E.5. The theoretical value is calculated for a sphere with a $\varnothing = 15\ldots20$ nm. Detailed measurements of *Alexa-488* are shown in Figure E.7. The theoretical value is taken from Ref. [173]. Detailed measurements of eGFP oligomers are shown in Figure E.9. The reference values are based on EMCCD data that were taken on the same instrument [119]. For the measurements where no binning was applied, the count rate per molecule is given.

6.8. Particle concentration measurements

Apart from the diffusion coefficient D, the second basic parameter that can be obtained from FCS measurements is the particle concentration c in the focal volume. To assess the performance of the CHSPAD sensor within the SPIM-FCS setup, several measurements of dilution series were performed on the instrument. For reference, the same samples were measured on a standard confocal setup in parallel.

Figure 6.18 shows the histograms of the measured concentrations c of a dilution series of *Alexa-488*. Every sample of the dilution series was clearly distinguishable from the others. The resulting histograms showed an almost GAUSSIAN shape, as expected for homogeneous samples, with an average relative width below $c_{vc} = 20\,\%$, which is comparable to the diffusion coefficient.

Figure 6.19 shows the measured concentration in comparison to confocal measurements of the same samples for the dilution series of three different samples, *Alexa-488*, *FluoSpheres YG* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads (*F8803*, INVITROGEN), and *QDot-525 streptavidin ITK* (*Q10041MP*, INVITROGEN). For the CHSPAD, each data point represents a single measurement. The mean value and the standard deviation were obtained for each sample from a GAUSSIAN fit of the histogram. In case of confocal FCS, each data point represents the mean value of at least six measurements. The coefficients that are presented in Table 6.4 were obtained from weighted linear fits (dashed lines in Figure 6.19). For the three different samples, linear relationships between the concentrations measured in a SPIM with the CHSPAD and on a confocal setup were obtained over almost three orders of magnitude. All three samples yielded concentration values of about a factor of 1.7 above the value obtained with the confocal setup. With the EMCCD camera, the overestimation on the same setup was typically three fold [119, p. 126]. The origin of the overestimation is not yet known but may stem from a too simple model for the focal volume in the model functions [119, 201].

The high temporal resolution of the CHSPAD allowed for a precise determination even of the fast *Alexa-488*, with approximately the same overestimation as obtained for the significantly brighter *TetraSpeck* beads. Due to the linear relation, this overestimation can easily be corrected for by a calibration sample of known concentration. Without further calibration, the setup is well suited for the assessment of relative concentrations.

sample	binning	$c_{\mathrm{SPIM}}/c_{\mathrm{confocal}}$
FluoSpheres YG beads	1×1	1.60 ± 0.02
Alexa-488	1×1	1.69 ± 0.13
	1×1	1.83 ± 0.06
QDot-525 streptavidin ITK	2×2	1.92 ± 0.07
	3×3	1.95 ± 0.08

Table 6.4.: Relative particle concentrations in the focus of various samples in relation to confocal FCS. The respective plots are shown in appendix E.

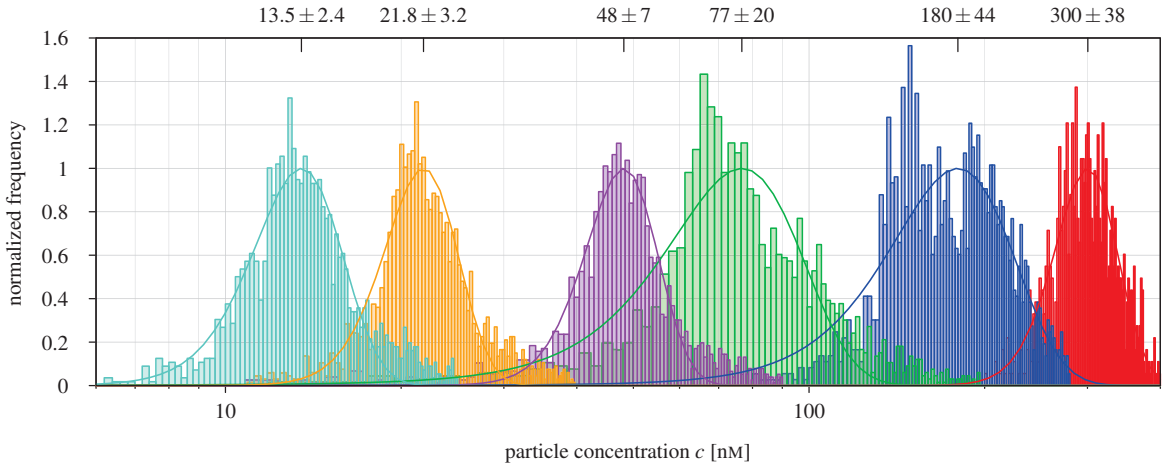


Figure 6.18.: ***Alexa-488*** **in solution: Histogram of the measured particle concentrations** c **for a dilution series obtained by a global fit.** For each sample, a GAUSSIAN fit is applied (solid curve); its mean value is written above the plot. Raw correlation estimates have been binned in a 3×3 region. Data is normalized to the maximum amplitude of the GAUSSIAN fit.

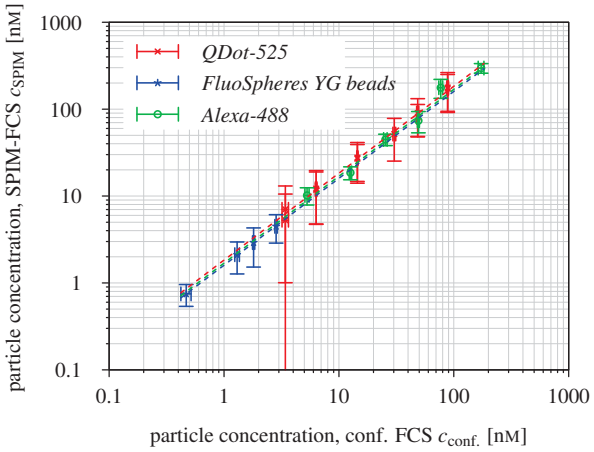


Figure 6.19.: Comparison of concentration measurements performed with the SPIM-FCS using the CHSPAD. Data is plotted as a function of concentrations, measured on a confocal FCS. A weighted fit is shown as a dashed line. Slopes are shown in Table 6.4. Error-bars indicate standard deviations. Data are corrected for an offset.

6.9. Mapping diffusion properties of eGFP-oligomers in HeLa cells

The cells for this section were prepared by Gabriele Müller (DKFZ, Heidelberg, Germany). Details on the cell culture protocols can be found in Ref. [119, pp. 206].

As described in the introduction (*cf.*, Figure 1.7), the final goal of this thesis was to create mobility maps of fluorescent proteins in live cells. In this section well established HeLa cells were used as a test system. These cells were genetically modified to produce different oligomers of enhanced green fluorescent protein (eGFP). Figure 6.20 shows a combined view of the transmission and fluorescence of the measured HeLa cells, expressing tetrameric eGFP. The image was acquired on an epi-fluorescence microscope. The measurements in this section were a proof of concept for further in-vivo measurements in live cells on the SPIM-FCS setup. Currently no other detection system is capable of such many simultaneous multi-spot measurements at such a high temporal resolution.

6.9.1. Measurement protocol

For live cell measurements, the protocol described in section 6.1 was adapted. The ROI on the sensor was chosen as small as possible, to still fit a single cell. Sample cells were selected by shape (no blebs, a recognizable nucleus, typical elongated shape, etc.), to measure healthy ones only. To be able to capture a high number of photons, only the brightest cells that could be found were measured. Raw data was captured until the 4 GB limit of the readout design was reached with a typical duration (depending on the exact ROI) of about 50 s or more. The data was bleach-corrected according to the procedure described in section 2.3.2 and the correlation curves were calculated using the correlator described in section 5.3. For further analysis, parts of the image outside of the cells were masked by imposing a threshold on the fluorescence intensity. As a last step, a global fit of all five correlation estimates (CEs) was performed as described in section 6.3.3 using the fit models described above.

6.9.2. Photobleaching

To account for the low sensitivity of the CHSPAD, again, a laser-power of approximately $p_{\text{eff}} = 20\,\text{mW}$ was used, which roughly corresponds to $I_{\text{LS}} = 260\,\text{W}/\text{cm}^2$ at the center of the lightsheet. This is about a factor of two higher than the laser-power typically used for cell measurements with the EMCCD camera on the same setup. The higher illumination intensity led to a strong photobleaching, which is shown in Figure 6.21 for a typical measurement. After a measurement, the cells in the focal plane were almost completely bleached and hardly visible. Free eGFP undergoes bleaching, too, but the pool of unaffected dye in a sample bag is some orders of magnitudes larger than in the volume of a cell. Therefore, in the sample-bags, un-bleached dye molecules constantly diffuse back into the focal volume and bleaching is significantly less pronounced.

Bleaching adds a time-dependent component to the correlation curves, which reduces the quality of the fits. This was accounted for by a correction of the raw data prior to the correlation according to equation (2.20).

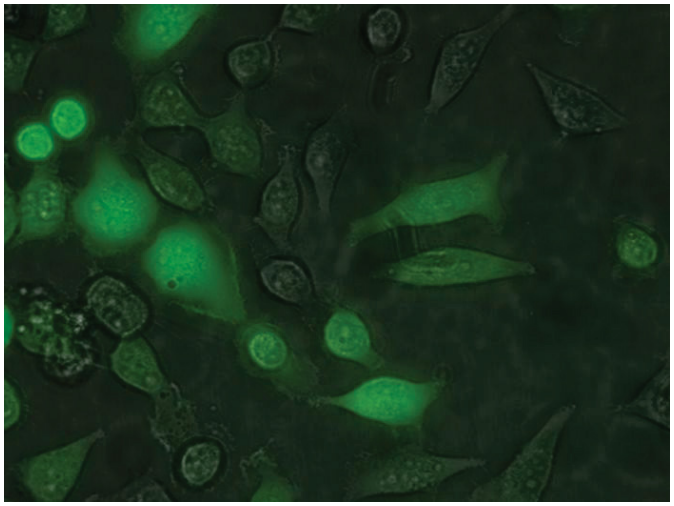


Figure 6.20.: Combined transmission and fluorescence images of HeLa-cells expressing eGFP-4x. Image width is 162 µm. The image was acquired on an epi-fluorescence microscope with a magnification of 40× and a LUMENERA *Infinity2-1R* CCD camera. The image have been optimized for better contrast. The image is taken from Ref. [119].

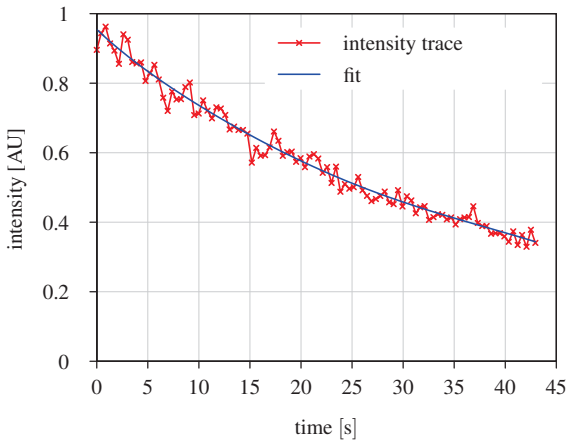


Figure 6.21.: Representative intensity time-trace of the fluorescence detected by a single pixel of the CHSPAD during a 43 s **measurement of eGFP-8x in HeLa-cells.** The intensity (red) decreases by a factor 2.7 over time. A fit of the model-function,equation (2.21), is shown in blue.

6.9.3. In-vivo measurements

Figure 6.22 shows the maps for three HeLa cells expressing different oligomers (monomers: eGFP-1x; tetramers: eGFP-4x; and octamers: eGFP-8x). The first column contains an overview image based on the detected fluorescence intensity. In the second column the map of diffusion coefficients D is shown. Finally, the last column contains a map of the particle concentration in the focus c. To impose no additional averaging, no binning was applied. Figure 6.23 shows the corresponding distributions of D and c including GAUSSIAN fits.

For the three cells, representative ACCs are plotted in Figure 6.24. The higher viscosity of the cytoplasm, in comparison to water (see section 6.6.2, for results from the free dye), reduced the diffusion coefficients of the eGFP oligomers approximately five-fold. This made it easier for the fit to distinguish between the afterpulsing component and diffusion component.

Figure 6.25 summarizes the results of several measurements of HeLa cells expression eGFP oligomers that were performed with the CHSPAD. Numbers are given in Table 6.5. For comparison, results for the same samples obtained on the same instrument with an EMCCD camera [119] and on a confocal FCS instrument performed in the same lab [59] are shown, too. Those results were obtained using a two-component model (for two diffusing species). Due to the higher noise and the afterpulsing in the correlation curves obtained with the CHSPAD, a two-component model could not be fitted reliably and results obtained with CHSPAD showed a significantly lower diffusion coefficient (almost factor of two). For the tetramers and octamers the measurements of the diffusion coefficients yielded a relative standard deviation of $c_{vD} \sim 25\,\%$ (with binning), which is about a factor of 1.5 above the value obtained for the free dye. In case of the much dimmer monomers, a relative standard deviation of $c_{vD} \sim 40\,\%$ was obtained. Without binning, the distributions were about two-fold wider.

To account for the systematic error introduced by a single component fit model, relative diffusion coefficients between two species were evaluated. These relative values were comparable for all three setups, showing that the setup with the CHSPAD allows to clearly distinguish between different eGFP oligomers. However, the high noise of the obtained maps did not allow for making statements on the spatial organization of fluorescent dye within the cell. For that purpose a more sensitive detector is required.

A rough estimation showed that the count rates per molecule for the samples plotted in Figure 6.23 were around 50 Hz. This value is significantly below the threshold of 300 Hz and explains the large width of the distributions. Again, as for the free eGFP in solution, the count rate per molecule did not change significantly for the different oligomers.

Sample	D_{CHSPAD}		D_{EMCCD}	D_{confocal}
	1×1 binning	3×3 binning		
$D_{20,\mathrm{W_{eGFP\text{-}1x}}}$ [µm^2/s]	20.7 ± 12.6	25.5 ± 9.9	38.2 ± 7.0	33.3 ± 7.3
$D_{20,\mathrm{W_{eGFP\text{-}4x}}}$ [µm^2/s]	8.6 ± 6.0	9.1 ± 2.4	15.5 ± 2.2	14.1 ± 3.3
$D_{20,\mathrm{W_{eGFP\text{-}8x}}}$ [µm^2/s]	5.6 ± 1.7	6.5 ± 1.6	11.6 ± 2.9	N/A
$D_{20,\mathrm{W_{eGFP\text{-}1x}}}/D_{20,\mathrm{W_{eGFP\text{-}4x}}}$	2.4 ± 2.2	2.8 ± 1.3	2.5 ± 0.6	2.4 ± 0.8
$D_{20,\mathrm{W_{eGFP\text{-}1x}}}/D_{20,\mathrm{W_{eGFP\text{-}8x}}}$	3.7 ± 2.5	3.9 ± 1.8	3.3 ± 1.0	N/A
$D_{20,\mathrm{W_{eGFP\text{-}4x}}}/D_{20,\mathrm{W_{eGFP\text{-}8x}}}$	1.5 ± 1.1	1.4 ± 0.5	1.3 ± 0.4	N/A

Table 6.5.: eGFP oligomers in HeLa-cells: Summary of the measured diffusion coefficients. Results given for the CHSPAD are median values of the measurements shown in Figure 6.25. EMCCD data was taken on the same instrument (*cf.* [119]). Confocal data was taken from Ref. [59]. The latter two used a two component model.

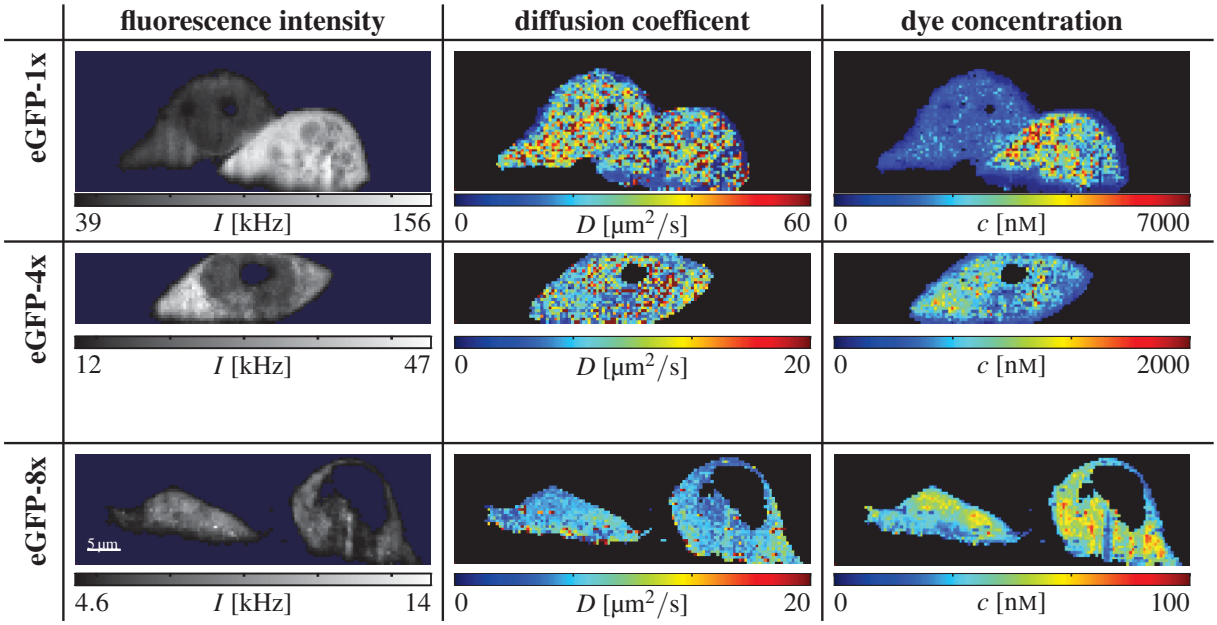


Figure 6.22.: eGFP oligomers in HeLa cells: Example maps of the diffusion coefficients and the particle concentrations. The first column shows the fluorescence intensity. The width of all images is 51.2 μm, the heights depends on the ROI. Data was obtained by global fits of the correlation curves.

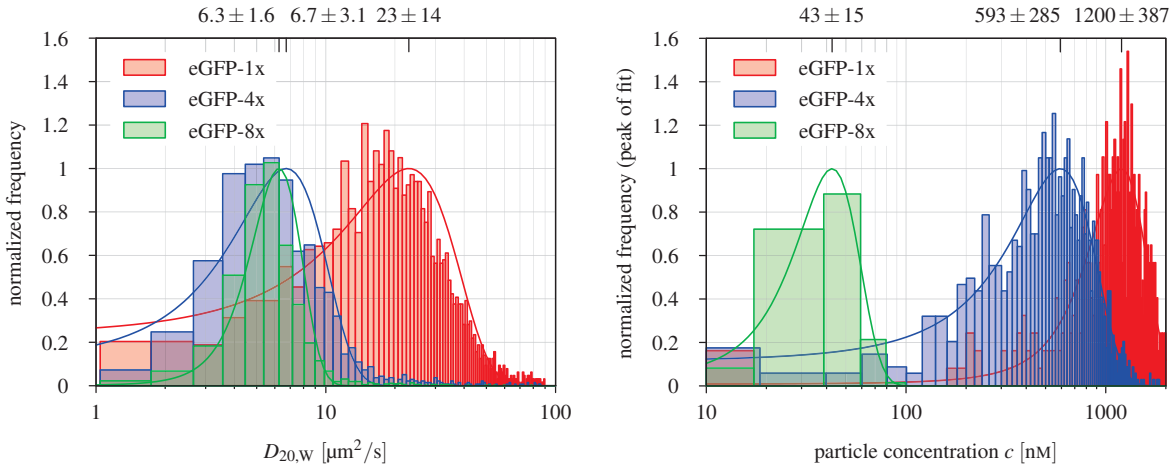


Figure 6.23.: eGFP oligomers in HeLa cells: Distribution of the diffusion coefficient D and particle concentration c in the cells shown in Figure 6.22. In case of eGFP-1x, only the inner part of the left cell was considered. Histograms are normalized to the maximum value obtained from GAUSSIAN fits. The corresponding mean values from the fits are shown above the plots. Raw data were not binned.

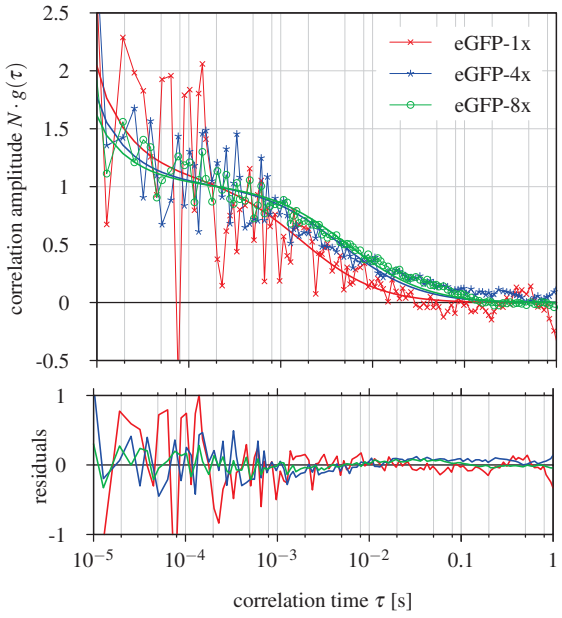


Figure 6.24.: eGFP oligomers in HeLa cells: Representative ACCs including global fits of the cells shown in Figure 6.22.

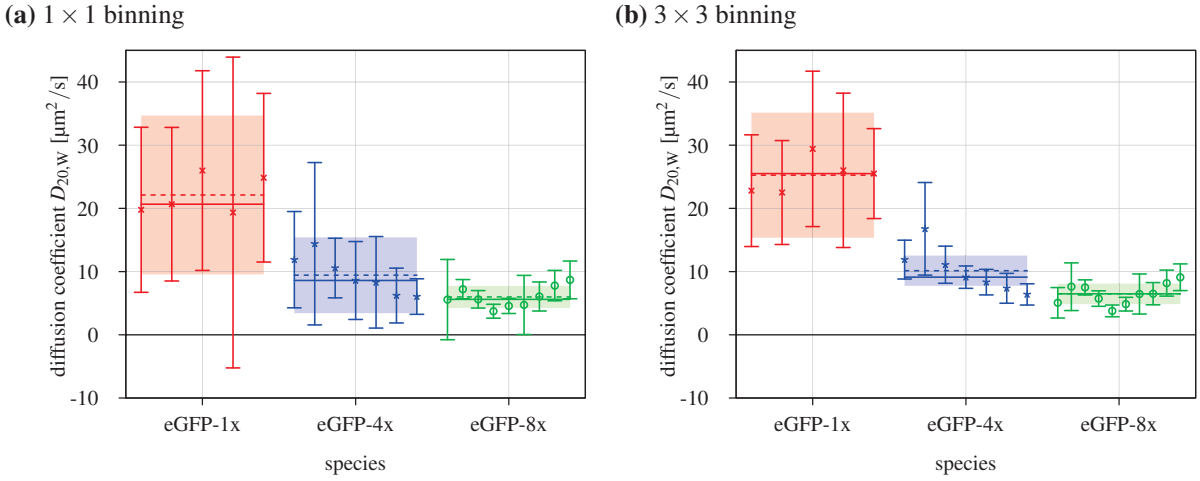


Figure 6.25.: eGFP oligomers in HeLa cells: Summary of the measured diffusion coefficients in different cells obtained by global fits. Results are shown without (a) and with 3 × 3 binning (b). Symbols are mean values from GAUSSIAN fits of the histograms of the measured diffusion coefficients. Error bars indicate standard deviations. The horizontal dashed lines represent the mean value, the solid lines the median value for all measurements of a single species. Semi-transparent areas indicate the median of the standard deviation for each species.

6.10. Conclusion

It can be stated that the CHSPAD with microlenses was used successfully as a sensor in a SPIM-FCS setup. The sensor performed best for bright samples (*e.g.*, *TetraSpeck* beads), but also less bright dyes used as biological markers were measured with good accuracy (*i.e.*, *QDot-525*, *Alexa-488* and eGFP oligomers).

Especially the use of a global fit of the auto- and cross-correlation curves proved advantageous. With this method it was possible to compensate for the inhomogeneity that was detected in the molecular detection efficiency. Moreover, the effect of afterpulsing that superimposed the ACCs could be mitigated so that the measurement of the diffusion of *Alexa-488* became possible. Global fits of the correlation curves allowed for a calibration-free measurement of various samples. The quality of the results was further improved by spatial binning of the correlation curves.

In addition to the diffusion coefficient also the particle concentration in the focus can be measured. Test show, that the results obtained with the CHSPAD tend to overestimate the concentration almost two-fold, when compared to measurements performed on a confocal setup. Thus, if absolute concentrations are to be measured, a calibration is required.

In the last section it was shown, that even in-vivo measurement were possible. Although the noisy data rendered the usage of typically used two-component fit models impossible, the diffusion of eGFP oligomers in HeLa cells could be determined. As the absolute values of the diffusion coefficient were influenced by the choice of the fit model and varied on different microscopes, the relative factors between any two samples were almost the same on all these instruments. Unfortunately, maps of diffusion parameters were to noisy to draw conclusions about the spatial distribution of the dye molecules within the cells. The overall low sensitivity of the CHSPAD with microlenses also required for an increased light intensity of the lightsheet, which also caused significant photobleaching. This was successfully accounted for by a raw-data bleach-correction.

7. Conclusion and outlook

7.1. Conclusion

This study investigated the suitability of new single photon avalanche diode (SPAD) arrays for imaging fluorescence correlation spectroscopy (imaging FCS). Two arrays were first evaluated in terms of general fitness for low-light imaging and especially for FCS-applications and appropriate readout-hardware was designed. Correlation algorithms required for data evaluation were then implemented and assessed on different high performance computing (HPC) platforms with regards to real-time image processing: FPGA, CPU, and GPU. Finally, both sensors were experimentally validated in an imaging FCS microscope.

7.1.1. Sensor hardware platforms

The first sensor evaluated was the *RADHARD2*. Developed in EDOARDO CHARBON'S lab (TU Delft, Netherlands and École polytechnique fédérale de Lausanne, Switzerland), the sensor was already available at the beginning of this work (mid 2010). Featuring 32×32 single SPADs, the sensor can be read out at a typical frame rate of 100 000 full frames per second. The readout of the array was done by a custom hardware design (see section 4.1), which was based on field programmable gate arrays (FPGAs). Raw data was transferred to the host computer via a standard USB 2.0 interface. Integrated into an existing lightsheet microscope, the *RADHARD2* was successfully used for the determination of diffusion properties of various samples. Lacking microlenses for efficient photon collection, the low photon detection probability (PDP) of this sensor ($< 1\,\%$) limited its application to bright samples (*e.g.*, fluorescent beads, see appendix E.4).

In 2012 a new sensor became available, the Swiss single photon avalanche diode array (CHSPAD), also from EDOARDO CHARBON'S lab. This sensor was integrated into the existing microscope setup, too, and appropriate readout electronics were designed. Being manufactured in a more recent CMOS process, the electronics in each pixel was shrunk and the pixel pitch was reduced from 30 µm to 24 µm. But still the sensor uses the same SPAD design as the *RADHARD2*. For full frames, comprising 512×128 pixels, a readout rate of up to 156 250 fps was achieved, which resulted in a total data rate of 1.2 GiB/s. Again, FPGAs were used for the development of a custom read-out design (see section 4.2).

To cope with the high data rate, two different approaches were used: The simple solution used the on board memory (4 GiB) of the sensor's mainboard to store the raw data. This limited the total measurement time to 3.3 s at full resolution, or longer for subregion read-out. After image acquisition, data was transferred to the host computer using a standard USB 2.0 interface, which usually took several minutes.

Real-time data acquisition was achieved using a second solution: Although the sensor board did not feature any fast interfaces, the bandwidth requirements were fulfilled using several IO-pins operated above their specification. The sensor's mainboard was connected with 20 differential pairs to a second FPGA board, which was then used as an interface-bridge to the host-computer's PCIe-interface.

Instead of optimizing the data transfer, one could also reduce the data rate. Simulations showed that lossless compression based on HUFFMANN coding could reduce the data rate more than two-fold (see section 4.2.6).

7.1.2. SPAD Arrays as detectors in SPIM-FCS

By the end of 2013, the CHSPAD array with microlenses finally became available for low-light measurements. The sensor was used to demonstrate live cell measurements on a SPIM-FCS microscope with commonly used dyes (*cf.*, section 6.9.3).

Due to their digital single-photon counting capabilities, SPAD arrays proved advantageous over electron multiplying charge coupled device (EMCCD) cameras for imaging FCS applications: In comparison to EMCCD cameras, the to-day work-horses for imaging FCS, the evaluated SPAD arrays offer a better signal to noise ratio (SNR) (see section 4.3.2). Additionally, no further noise is added during the readout process or during additional amplification steps in the gain-register (EMCCDs only, so called excess-noise).

When looking at the number of pixels, their amount on the SPAD arrays was comparable to state-of-the-art EMCCD cameras, which are typically used for lightsheet fluorescence microscopy (LSFM), [26, 123, 200]. The two orders of magnitude higher frame rate allowed for measuring molecular motion that cannot be resolved otherwise (*e.g.*, small-molecules used as drug targets [71]).

The main factor that limited the usability of the CHSPAD for low-light fluorescence correlation spectroscopy (FCS) measurements was the overall low PDP. Although the quantum efficiency (QE) is about 40 %, and comparable to scientific cameras, the low fill-factor significantly reduces the PDP of each pixel to 2.2 %. In comparison, back illuminated EMCCD cameras have a PDP above 90 %. This could partly be accounted for by using microlenses, which increased the PDP approximately ten-fold.

A second limiting factor of the evaluated SPAD arrays was a significant afterpulsing. For the CHSPAD array, the decay time of the afterpulsing was about one order of magnitude slower than expected for substrate related effects. The afterpulsing affected several orders of magnitude of lag-times in the autocorrelation curve (ACC), and could not be separated from the actual signal, especially if the detected signal is low. The exact origin of this afterpulsing-like component could not be identified exactly during this thesis (*cf.*, section 4.2.4).

The afterpulsing disappeared in the spatio-temporal cross-correlation. Though, by using a global fit, the advantages of afterpulsing-free cross-correlation estimates and the information content of the autocorrelation estimate could be combined for more precise estimation of the diffusion coefficients. Additionally, cross-correlation introduces an independent “ruler”, as the pixel distance is precisely known. In combination with the autocorrelation estimates, this allowed for calibration-free measurements. However, the spatial resolution is lowered when evaluating the cross-correlation curves, because multiple pixels are taken into account and averaged over.

Global fits were also used successfully to compensate for an inhomogeneity that was detected in the molecular detection efficiency (MDE) of the SPIM-FCS setup. Measurements showed, that the focal volume at each pixel was not constant as expected. By using a global fit these values could be used as free parameters, and the diffusion could be determined more accurately (*cf.*, section 6.3.3).

A major drawback of the CHSPAD sensor platform was the lack of temperature stability. Usually, commercially available scientific cameras adapt active cooling to keep the sensor’s temperature constant. The combination of sensor board and motherboard did not allow for any efficient cooling of the sensor. Depending on the incident light intensity, a sensor temperature of up to 50 °C was reached (see section 4.3.9). This heating significantly increased the dark count rate (DCR) and active or passive cooling will prove advantageous for any further application of the CHSPAD.

7.1.3. Correlation analysis

Beside FCS, correlation analysis is used to improve SNR of receivers (*e.g.*, RADAR or GPS). In FCS, this method is used to extract diffusion properties of the samples. Thus, several orders of magnitude need to be evaluated in the time domain.

Correlation analysis for imaging data can either be done pixel-wise for a single color channel (autocorrelation), for two pixels but one color channel (spatio-temporal cross-correlation), or for two color channels (fluorescence cross-correlation spectroscopy, FCCS). For diffusion measurements, typically the first two methods are applied. FCCS can be used to distinguish differently labeled species or to evaluate reaction kinetics. The large amount of pixels and the high frame rate offered by the CHSPAD sensor required for fast algorithms.

Signal convolution, often used for digital image processing, is closely related to correlation analysis (WIENER-CHINTCHIN-theorem). Thus, beside the direct evaluation of the correlation integral with, for example, a logarithmic spacing (see section 2.4.2), also data evaluation based on fast Fourier transform (FFT) is possible (*cf.*, section 3.2). Both variants, direct and FFT-based correlation, have a computational complexity of $\mathcal{O}(N \cdot \log N)$ and rely on the full raw dataset, with N samples. To overcome the limit of memory size, imaging FCS correlation analysis is typically done using a correlation algorithm which allows for streaming.

As several orders of magnitude of lag times τ are evaluated, the computational costs for a simple linear spacing of the τ-values would be too high ($\mathcal{O}(N^2)$). Thus the multiple-τ correlation algorithm, based on a cascade of short linear correlators (so called blocks), is employed. For each subsequent linear block the frequency of execution is reduced, creating a quasi-logarithmic scale. Typically, consecutive input values for each block are accumulated to not loose information. The implicit averaging that increases from block to block is beneficial for the application of FCS as the correlation decays to zero. On the contrary, for signals that do not decay (*e.g.*, a sine wave), the multiple-τ algorithm is counter-productive. As the correlation function is evaluated at a constant number of fixed time points, the complexity is reduced to $\mathcal{O}(N)$.

The multiple-τ correlation algorithm, that was used in this work, allows for easy parallelization (see section 2.4.2). This can either be done trivially on the level of pixels, or even on the level of linear blocks. As all linear blocks are equal, except for their time basis, the hardware for a single block, or even a single lag, can be re-used to form a full correlator. If performance requirements permit, such hardware can even be reused for multiple pixels.

In this study, several implementations of the multiple-τ correlation algorithm were explored in chapter 5 with regards to a massively parallel correlation analysis of more than 1000 pixels and real-time performance. The explored implementations were based on different hardware platforms, a XILINX VIRTEX 2 FPGA, two current CPU models, and a recent consumer-grade graphics processing unit (GPU). On all three platforms, the correlators are centered around fast multiply accumulate (MAC) operations (DSP-slices on FPGAs, FMA operations on CPUs and GPUs). So far, available software and hardware implementations (*cf.*, chapter 3) were either optimized for a few pixels only or did not achieve real-time processing for the data rate of the CHSPAD.

For the CHSPAD sensor, the aim of real-time autocorrelation analysis was achieved on the GPU platform (see section 5.4). For both, the CPU and the GPU, the achieved performance was significantly higher than any other implementation reported so far (*cf.*, section 3.2). Details on the three different implementations are illustrated in the following.

GPU correlator

The GPU implementation of the multiple-τ correlation algorithm offered the highest performance of all platforms (approximately 420 Gflops/s). Being 2.6× faster than real time on a consumer-grade NVIDIA *GTX 780Ti* GPU, the correlator outperformed existing implementations by several orders of magnitude.

For the implementation of the GPU correlator, the NVIDIA compute unified device architecture (CUDA) SDK was used. Thus, the code can be run on any NVIDIA GPU that supports compute capability 2.0 or higher. The correlator was implemented making use of the massive parallelism offered by the GPU. Therefore, for each pixel a single thread was used to correlate the data (SIMT architecture).

The used algorithm exploits the large amount of graphics memory and the large number of registers in order to reduce the frequency of context switches. Thus, the most limiting factor of the implementation turned out to be the memory-bandwidth of the graphics memory. Further improvements were described, that should allow for a speedup of approximately 3.4× over real time. Still being limited by the memory-bandwidth, that way $\sim 22\,\%$ of the theoretical peak performance of the GPU should be achieved.

CPU correlator

The implementation of a CPU correlator was done using the current SIMD instruction set extensions and recent fused multiply accumulate (FMA) operations. On the CPU platform, based on the INTEL *Haswell* microarchitecture available in this work (INTEL *4770*, quad-core processor), $50-60\,\%$ of the required performance for real-time correlation analysis could be reached. This was approximately $39\,\%$ of the theoretical peak performance of the chip, which is similar to results obtained by the *LinPack* benchmark [110]. On the second platform, based on the AMD *FX-8320* CPU, approximately $30\,\%$ of the real-time performance was achieved.

As shown in section 5.3.5, the algorithm scaled linearly with the number of cores, so replacing the CPU by a model with eight cores (*e.g.* the consumer-grade INTEL *5960X* CPU), real-time correlation should be achievable. If double precision floating-point values instead of single precision floating-point values were used as data type, which is particularly required for bleach-corrected data, the performance was halved.

Following the GPU implementation, the CPU correlator is the second fastest multiple-τ correlator offering $85\,\mathrm{Gflops/s}$. With that, the CPU correlator was significantly faster than other CPU-based implementations.

FPGA correlator

The implementation of an FPGA correlator for the *RADHARD2* was the first that was able to deal with more than 1000 pixels. Based on hand-written hardware description language (HDL) code, real-time correlation was achieved at a temporal resolution of the sensor of $10\,\mu\mathrm{s}$ [155]. The design is based on 32 correlators running in parallel, each handling 32 pixels. Its performance was comparable to other implementations based on the same XILINX VIRTEX 2 family (section 3.2). Naturally, the presented implementation is outperformed by more recent hardware and did not achieve the performance of commercial correlators, although it offered significantly more channels. Assuming a moderate 25× increase in performance (clock speed and logic resources) for a recent XILINX VIRTEX 6 FPGA over the VIRTEX 2 platform, the same performance as the fastest proposed hardware design should be achievable (*cf.*, section 3.2).

Dataflow computing

Contrary to classical computing, dataflow computing does not use an imperative programming model to model an algorithm but describes it as a directed graph. In such a graph data flows between small processing elements, which perfectly map massively parallel hardware, such as FPGAs.

In section 5.5.7 a correlator design based on such a dataflow description was proposed. This design should allow for an efficient implementation on dedicated dataflow computing platforms as, for example, MAXELER TECHNOLOGIES *MaxStation*. Within the limits of such hardware (recent FPGAs), real-time correlation should be feasible. A major benefit of this approach is the significantly reduced development-time compared to hand-written HDL code, although typically a full redesign of the algorithm is required.

Cost and time efficiency

From a financial point of view, the cheapest correlator was the GPU-implementation, which also offered the best performance. The INTEL *Haswell 5960X* costs about a factor of two more than the GPU, but did not provide comparable performance for this type of algorithm. At least another factor of two has to be budgeted for the acquisition costs, if an FPGA-based solution is chosen. However, an FPGA is required anyways for the SPAD readout, so this one can also be used for correlation, if free resources remain.

From the developer's perspective, the FPGA implementation, based on hand-written HDL, required the longest development time. Although the core component with the digital signal processor (DSP) block was implemented within days, much effort had to be put into its optimization and the design of the periphery (memory controllers, USB interfaces, *etc*.). The development of a CPU-based correlator was about two to three times faster. Here, a time-consuming part was the creation of platform-independent code, because several CPUs were used for the evaluation of the algorithms. Although a special software library was utilized for platform-independent vectorization, support for specific features of each CPU had to be implemented manually. Even less time was spent for the GPU platform.

A main advantage of the CPU or GPU software is the much faster response to changes in the design: Source code can be compiled and executed within seconds (in contrast to tens of minutes for HDL-code synthesis for FPGAs). The same applies for the debugging process: On the CPU and GPU well known debuggers can be used and dumping the contents of variables is done easily.

7.2. Outlook

7.2.1. CHSPAD real-time data acquisition and correlation analysis

In a typical confocal setup with two avalanche photodiodes (APDs), the detector signals are directly fed into a correlator-card. This device digitizes the raw signal and performs the correlation analysis of the signals, typically two autocorrelations and two cross-correlations. The estimated correlation functions are displayed in real time so that they can be used for alignment (*cf*., Ref. [200]) or immediate inspection of the sample quality.

In imaging FCS it would be beneficial to display real-time maps of the ongoing diffusion, which could be used for the alignment, too (*i.e*., optimizing the size of the focal volume of each pixel). Depending on the model, the required curve fitting typically takes several minutes for the CHSPAD sensor on a recent CPU. First efforts were made to bring the LEVENBERG-MARQUARDT fit algorithm used in here onto the GPU have been proposed in Ref. [259]. The gain in speed compared to a quad-core CPU was reported to be about 50× and may allow for almost real-time estimation of diffusion parameters.

Real-time operation of the microscope also allows for choosing samples with a good quality. So far, cells were selected based on the visible fluorescence intensity. If in parallel to the intensity image a false color image of the diffusion parameters was provided, the selection process of the samples could be accelerated. Additionally, no further time is spent waiting for the results of the correlation analysis. Such a selection process saves expensive disk space and evaluation time, because samples that do not meet the requirements can be excluded prior to evaluation. If the raw data were recorded in parallel, too, a more in-depth analysis could be done afterwards, *e.g*., bleach correction or further cross-correlations.

Another limiting factor of the microscopic setup was the hardware platform for the CHSPAD sensor itself, which lacked any high-speed interfaces. Thus, real-time data transfer of the raw data was only possible when the hardware was used above specifications. To solve this issue, a new platform is under development* featuring a dedicated, up-to-date FPGA for readout and high-speed interfaces (*i.e*., USB 3.0 or better). The new platform will also allow for efficient cooling of the sensor.

*Based on personal communications with the designer of the CHSPAD, SAMUEL BURRI.

7.2.2. Correlator development

The trend in computer industry towards more parallelism, which especially applies for graphics cards, is ideal for imaging FCS. Since the autocorrelation analysis is independent for every single pixel, more parallel computing hardware allows to process more pixels. According to GUSTAFSON'S law, a linear scaling of the algorithm is expected, and future arrays in the mega-pixel range can be handled, too. If the computing hardware is not used to full capacity for autocorrelation analysis, even cross-correlation curve can be calculated simultaneously.

A still limiting factor for correlators based on the streaming algorithm (see section 5.2.3) is the memory bandwidth of current CPUs and GPUs. With new three-dimensionally stacked memory placed directly on-chip, these limitations were addressed. GPUs featuring this new architecture were recently released by AMD [6]. A similar GPU-architecture with on-chip memory was announced by NVIDIA, too [164]. With up to 1 TB/s of bandwidth and more than 20 GB of memory, these architectures should be ideal for high performance computing in general and especially for correlation analysis.

The correlation algorithms still offer potential for additional optimizations. In case of the GPU implementation, an implementation was proposed that can further reduce execution times. Also the FPGA-based correlator did not make use of the full potential of the underlying hardware: By taking advantage of the binary nature of the input signal, the first linear blocks of the multiple-τ correlator could be implemented as shift registers with a trivial 1×1 bit multiplication. This helps to increase the clock rate and to save DSP resources of the correlator. A similar but less performant solution was already described in Ref. [72].

7.2.3. SPAD arrays

The temporal resolution of the detector (*i.e.*, its frame rate) is crucial to resolve fast molecular motion of small particles. In the used SPIM-FCS setup, the decay time of small molecules is typically of the order of 100 µs (*e.g.*, *Alexa-488*: $\sim$ 150 µs). The frame rate has to be at least one order of magnitude higher than the decay time to resolve the plateau of the correlation function, which contains information about the particle number. Especially when only the occurrence of one photon (single bit memory) can be stored in each pixel, a higher readout rate reduces the probability to miss photon events. Therefore, the CHSPAD sensor was read out at maximum speed in rolling shutter mode.

Unfortunately not all external signals are applied row-wise to the individual diodes. Some of them apply to the entire sensor. Especially when active recharge is used (manual re-arming of the SPADs), it would be advantageous to have row-wise control, as for example of the reset-signal. Then, the active period of every single SPAD would be identical and as long as possible.

On the other hand, the gating signals, to turn on and off the SPADs are not required for imaging FCS and could be removed in favor of larger diodes, which improves the fill-factor and the photon detection probability. The overall limited fill-factor could be increased by further reducing the size of the electronics. Another option to gain sensitivity is to use rectangular-shaped diodes or back-illumination, as used for EMCCD cameras.

The aspect ratio of the CHSPAD array is well suited for multicolor setups. Here, the intensity signal from the sample is typically split into several color-channels that are imaged onto the sensor simultaneously side-by-side. As SPIM-FCCS is well established using cameras, the application of SPADs is the next step. The aspect ratio of the CHSPAD is particularly suited for multi-color setups.

7.2.4. FCS measurements

Light-induced bleaching of the sample was a major problem of real-time FCS. By adding a component on long time scales to the intensity signal, the correlation curves get distorted, rendering any model-fitting impossible. Bleach-correction algorithms are based on modifying the raw data, when the progression of

the intensity is known. As this is only known after the measurement has ended, a correlation analysis cannot be done in parallel. Bleach correction is applied to the binary input data stream, and requires floating-point values. In case of an FPGA-based correlator such data are difficult to handle, as floating-point values are not supported natively. Moreover, single precision floating-point values do not provide sufficient accuracy, making a re-design of the correlators based on double precision mandatory.

Therefore, algorithms are required that allow post-correlation corrections for bleaching. If, for example, an intensity trace is required for each pixel, this can be generated by the acquisition hardware in parallel with a significantly lower temporal resolution.

Appendices

A. Materials and methods

A.1. SPIM microscope

description	manufacturer	comments used settings
power-meter	Thorlabs *PM100D* with *S121C* detector (400 nm – 1100 nm, 500 milliW)	$\lambda = 491$ nm, auto-range
EMCCD camera	*iXon X3 860 iXon X3 860*	
TetraSpeck $\varnothing = 100$ nm fluorescent beads	INVITROGEN, *T7279*	
FluoSpheres YG $\varnothing = 100$ nm fluorescent beads	INVITROGEN, *F8803*	
QDot-525 streptavidin ITK	INVITROGEN, *Q10041MP*	
Alexa-488	INVITROGEN	

Table A.1.: List of materials and components used during the experiments.

A.2. Computers

For the development of the CPU correlator on the INTEL and AMD platform, the two following computers were used.

B040-SPIM2

component	details
mainboard	GIGABYTE *GA-990FXA-UD3*, BIOS FD 05/30/2012
CPU	AMD *FX-8320* eight-core processor
memory	2×8 GB KINGSTON *KHX18C10T3K2/16X*, DDR 3 1600 MHz
GPU	NVIDIA *GTX 780Ti*
harddisk	WESTERN DIGITAL *WD2002FAEX*, 2 TB

Table A.2.: *B040-SPIM2* hardware specifications.

software	details
gcc compiler	4.7.3 (Gentoo 4.7.3-r1 p1.4)
Linux kernel	3.10.25-gentoo
Vc (section 2.6.4)	0.7.1 (released on 26 Mar 2013), including various patches
NVIDIA CUDA	5.5.22

Table A.3.: *B040-SPIM2* software specifications.

B040-SPIM3

component	details
mainboard	ASUS *Z87-A*, BIOS 1007
CPU	INTEL *i7-4770* [45]
memory	2×8 GB KINGSTON *KHX1600C10D3B1K2/16G*, DDR 3 1600 MHz
GPU	integrated into CPU
harddisk	WESTERN DIGITAL *WD1002FAEX*, 1 TB

Table A.4.: *B040-SPIM3* hardware specifications.

software	details
gcc compiler	4.7.3 (Gentoo 4.7.3 p1.1)
Linux kernel	3.10.25-gentoo
Vc (section 2.6.4)	0.7.1 (relaesed on 26 Mar 2013), including various patches

Table A.5.: *B040-SPIM3* software specifications.

B. FPGA development boards

B.1. LASP

The *LASP* FPGA development board (EPFL, Switzerland) was used as mainboard for the *RADHARD2* SPAD array. Table B.1 gives an overview of the main resources.

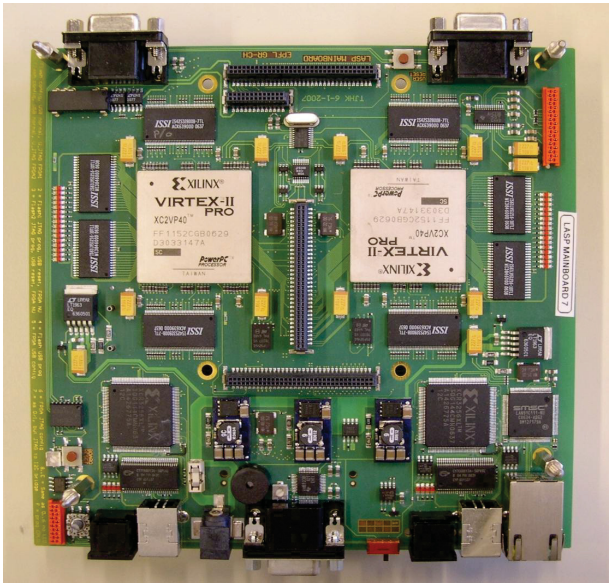


Figure B.1.: Photograph of the *LASP* development board. Image taken from Ref. [36].

resource	details
FPGA	2× VIRTEX-II *XC2VP40*
SRAM	2×4 ISSI IS61LV51216, 10 ns, $512k \times 16$
SDRAM	2×2 MICRON MT48LC2M32B2, *PC-100* SDR-SDRAM, $2\,\mathrm{M} \times 32\,\mathrm{bit}$
USB interface	2× CYPRESS *CY7C68013A* EZ-USB FX2 USB Microcontroller High-Speed USB Peripheral Controller

Table B.1.: *LASP*: Main hardware resources.

B.2. *Broaddown 4*

The *Broaddown 4* FPGA development board (Enterpoint Ltd., Malvern, United Kingdom) was used as mainboard for the CHSPAD sensor. Table B.2 summarizes the main resources used in this work. Refer to the manual for further details (Ref. [137]).

resource	details
FPGA	2× VIRTEX 4 *Virtex4 XC4VLX100-10*, see [246] for an overview
SDRAM	CRUCIAL *CT51264AC800*, 4 GiB, DDR 2-SDRAM (configured as MICRON *MT16HTF51264HZ* in the controller core)
USB interface	CYPRESS *CY7C68013A* EZ-USB FX2 USB Microcontroller High-Speed USB Peripheral Controller
high-speed clock generator	1× *ICS8442* and *ICS8745 PLL* up to 700 MHz
low speed clock generator	2× *cypress CY22394*

Table B.2.: *Broaddown 4*: Main hardware resources.

High-speed link between the development boards

The high-speed link reaches from the 'Edge Connector' of the *Broaddown 4* to a *FMC XM105 Debug Card* attached to the XILINX *ML605*. Table B.3 summarizes the hardware.

resource	details
Broaddown 4 side	3M *7860-0000PR* PCB Connector Grid pitch: 2.54 mm, 60 pins
cable	3M Volition Cat.6A S/FTP 1 m
XILINX *ML605* side	no name PCB Connector, 2.54 mm pitch, 40 pins

Table B.3.: Hardware used for the link between the FPGA boards.

C. SPIM: light intensity measurement

C.1. Light intensity of the lightsheet

Figure C.1 shows the light intensity at the center of the lightsheet as a function of the laser power. The height of the lightsheet was 1.5 mm; the width 2.7 µm.

C.2. LED light source

For the majority of measurements, a custom built LED light source was used [119], which can be remotely controlled with *QuickFit3*. Figure C.2 shows the dependence of the output power on the settings used. A fit shows that the LED light source can be assumed linear within the used range.

C.3. EMCCD as reference

For all measurements with the CHSPAD, the EMCCD is used as reference. It accounts for slight changes in the illumination path, the light path, or the sample chamber. Figure C.3 shows the intensity measured by the EMCCD camera as a function of the set current flux through the LED light source. A linear fit shows the expected linearity.

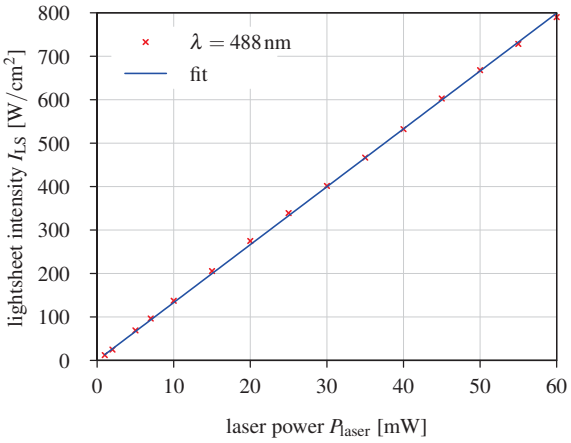


Figure C.1.: Intensity of the lightsheet in relation to the output power of the 491 nm **laser.** No additional ND filter was integrated into the light path. Slope from fit: 13.3. Data taken from Ref. [119].

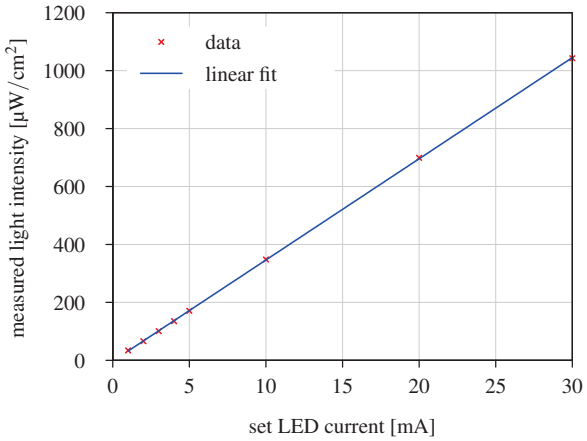


Figure C.2.: Light intensity of the LED as a function of the set current. Output power, as measured with a power-meter, as a function of the set LED current. Raw data is marked with +, the red line resembles a linear fit.

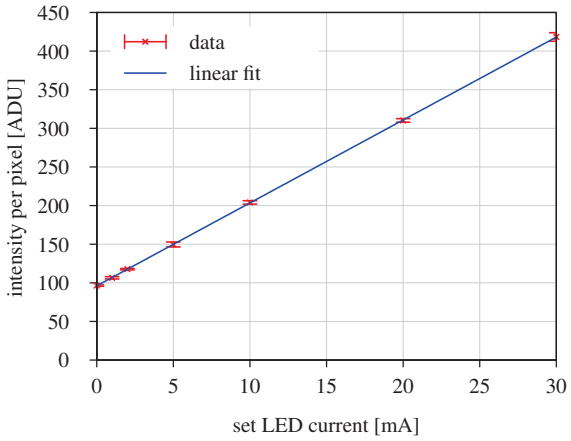


Figure C.3.: Measured intensity of the EMCCD camera depending on LED light source current. Raw data is marked with +, the red line resembles a linear fit. Error bars indicate the standard deviation.

D. CPU correlator: efficient data conversion

A different approach that tuned out to be faster is based on logic operations using floating-point and integer instructions in combination on the same values. Listing 4 shows the source code of the algorithm. First, the raw data is then loaded to an MMX register and copied into every entry with a single instruction (line 7). Using a constant binary mask with only one bit set per entry, the raw data is tested for set bits (line 8) using an `and` operation. Although still integer data is in the register, a further conversion step from the resulting integer to the float data type (`_mm256_cvtepi32_ps`, between line 8 and 9) can be omitted, as the representation of 0 are identically for both data types (floating-point: exponent $E = 0$, mantissa $M = 0$). The comparison with 0 yields a vector-mask (line 9) which then in turn can be used to select either ones or zeros (line 10). Thus the entire conversion can be done in four SIMD-instructions. Typically, more than one conversions are done serially so the constants can reside in the register file.

```
1   const __m256 val_0=_mm256_setzero_ps();
2   const __m256 val_1=_mm256_set1_ps(1.0F);
3   const __m256 val_c=reinterpret_cast<__m256>((__m256i){0x0000000200000001,
      ↪ 0x0000000800000004, 0x0000002000000010, 0x0000008000000040};
4
5   inline __m256 convert(uint8_t *data){
6     register __m256 ymm;
7     ymm = _mm256_broadcast_ss(reinterpret_cast<const float*>(data));
8     ymm = _mm256_and_ps(ymm, val_c);
9     ymm = _mm256_cmp_ps(val_0, ymm, 0x00);
10    return _mm256_blendv_ps(val_1, val_0, ymm);
11  }
```

Listing 4: Raw input data conversion using AVX. 1 Byte of input data is converted to a vector of 8 single precision floating-point values.

E. Further SPIM-FCS measurements

In this section, further SPIM-FCS measurements of different samples obtained with the selective plane illumination microscopy (SPIM) using the CHSPAD sensor with microlenses are shown.

E.1. Inhomogeneity of the size of the focal volume

To assess the above described inhomogeneity of the focal volume, additional FCS-measurements of *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads were performed with slight variations in the alignment of the microscope. The parameter of choice was the overlap of the lightsheet and the focal plane of the detection objective, which is tuned by the gimbal mounted mirror (GMM). For the evaluation, the diffusion coefficient D in the fits of the ACCs was fixed, whereas the lateral half width of the focus GAUSSIANw_{xy} was used as a free parameter.

Figure E.1 shows the dependency of w_{xy} on the position for three different setting of the overlap. If combined, the three plot show a parabolic shape in the course of w_{xy} along the x-axis. The value of w_z was estimated from a bead-scan. In between the measurements the GMM was rotated slightly by turning the micrometer screw for a few degrees. To preclude any effect of the microlenses, a comparative measurement was performed without microlenses, which is shown in Figure E.2. Although the obtained shape of the curve is more distorted, the same parabolic shape is detectable.

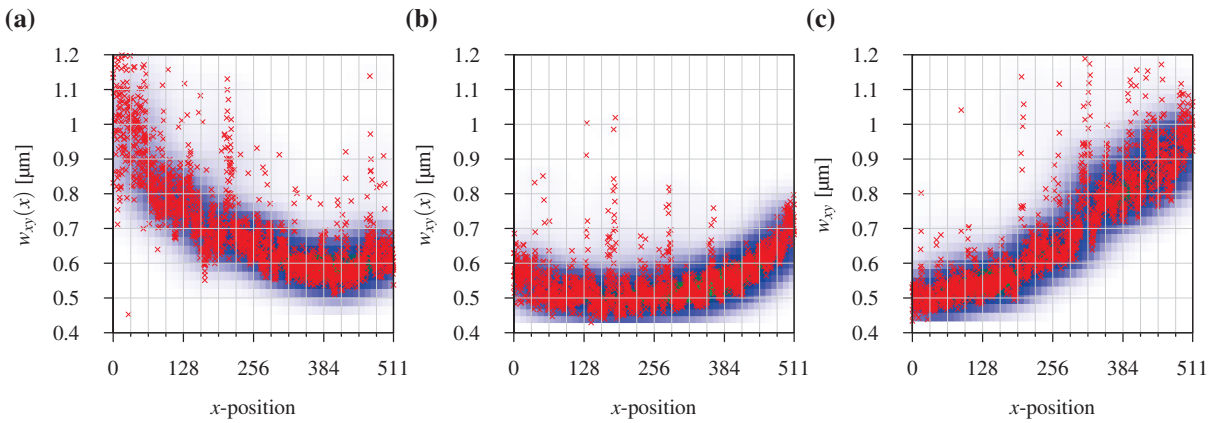


Figure E.1.: Dependence of w_{xy} on the x-position for a *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads sample. The ACCs were fitted with w_{xy} as a free parameter and a fixed diffusion coefficient taken from global fit. Three different settings for the manual alignment of the overlap of the lightsheet and the focal plane of the detection objective were chosen: (b) best possible alignment (maximum sharpness); (a) slightly de-focused. (c) also slightly de-focused, but to the other direction. In the background a GAUSSIAN kernel density estimation is shown. Faulty pixels were excluded. Outliers are most likely due to aggregates passing by.

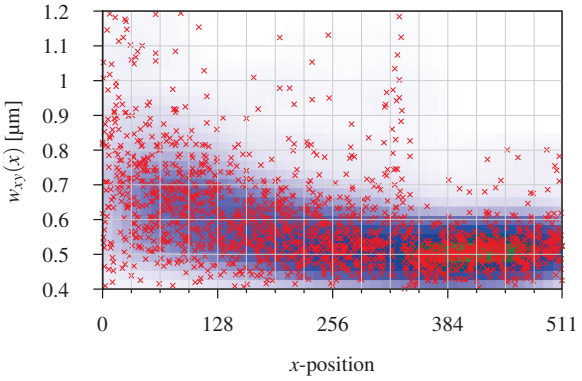


Figure E.2.: CHSPAD without microlenses: horizontal dependency of lateral half width of the focus GAUSSIANw_{xy} for *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads. The ACCs were fitted with w_{xy} as a free parameter and a fixed diffusion coefficient ($w_{xy} = 1.5\,\mu\text{m}$). $P_{\text{eff}} = 60\,\text{mW}$.

The plots suggest that a slight misalignment between the focal plane of the detection objective and the lightsheet is present, which was not unveiled by the bead-scans (see section 4.3.8). This effect was not detectable by the EMCCD camera, where w_{xy} can be considered constant in a 128 pixel wide region.

Conclusion

It appears, that by slightly shifting the lightsheet along the y-axes, the minimum of the w_{xy} distribution could be moved along the x-axes of the sensor. This optimization of the overlap of the focal plane of the detection objective and the lightsheet is the last step of the alignment of the instrument (*cf.* protocol) Typically this is done by manually focusing *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads using the EMCCD camera. When a good overlap of both the lightsheet and the focal plane is reached, w_{xy} is minimized and the sharpness of the image is enhanced. Focusing the sample manual works quite well with rather large and bright fluorescent sample, as for example 100 nm fluorescent latex beads, and is rather impossible for small dyes like *Alexa-488*. Possibly, an automated optimization based on the calculated sharpness parameter proves advantages. For that purpose, the micrometer screws of the GMM used for alignment have to be replaced by piezo-electric ones.

The possibility to move the minimum suggests, that both planes are not perfectly parallel but do cut under a small angle. A further position dependence of w_z was not evaluated so far.

E.2. Distribution of the diffusion coefficient obtained by global fits of different sets of correlation curves

In Figure E.3 the histograms over the diffusion coefficients obtained for all pixels using different sets of correlation functions estimates are shown. The histogram for the *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads beads described above (run 5) is shown in Figure E.3a. A different measurement of *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads beads is shown in Figure E.3b. In Figure E.3c a different kind of beads of the same size were used (100 nm yellow-green (YG)). The histograms of the diffusion coefficients obtained for every single pixel using the results of an individual autocorrelation analysis is shown in red, the histograms in blue and green were obtained from fits of the cross-correlation curves (CCCs), for the next neighbor (blue) and two pixel distance (green). In case of the CCC 2, w_{xy} was used as a free parameter. The histograms of the diffusion coefficients obtained by a global fit (autocorrelation curve and left and right cross-correlation curve with one and two pixel distance) are shown in magenta. To not rely on a specific alignment, all measurements were carried out on different days. A GAUSSIAN fit of each histogram is shown as a solid curve. Two examples of the corresponding correlation curves for a single pixel including global fits are shown in Figure 6.6.

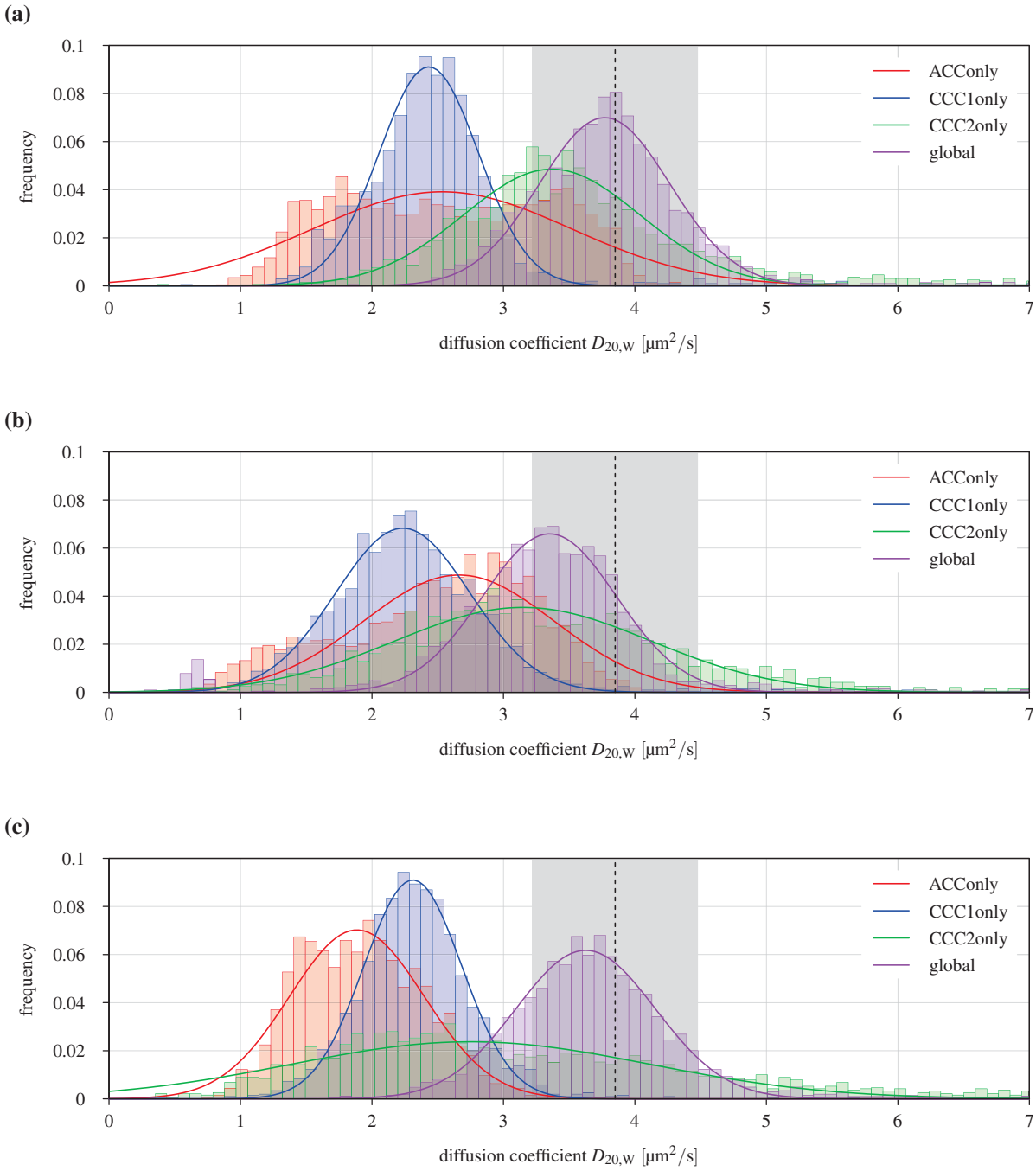


Figure E.3.: Example of histograms over the diffusion coefficients D of two samples of 100 nm beads in water using global fits of different sets of correlation estimates. A GAUSSIAN fit (solid curve) is applied to each histogram (not all are shown). The suffix 'only' denotes a fit of a single correlation estimate with $w_{xy} = 1348$ nm (from bead-scan, see Table 6.1). 'ACC +CCC 1' denotes a global fit of the ACC and the left and right CCC. In a global fit, w_{xy} was used as a free parameter. In case of cross-correlation estimates, both, left and right neighbors were taken into account. The curves of CCC 2 and CCC2only did not decay to zero, therefore an offset-correction was added to the model function. For the CCC2only fits, w_{xy} was used as free parameter. (a) *TetraSpeck* $\varnothing = 100$ nm fluorescent beads as already shown in Figure 6.1b and Figure 6.2. Result from global fit: $D_{20W,\,\text{global}} = (3.77 \pm 0.50)\ \mu m^2/s$. (b) *TetraSpeck* $\varnothing = 100$ nm fluorescent beads beads with $I_{LS} = 27\,W/cm^2$. Result from global fit: $D_{20W,\text{global}} = (3.35 \pm 0.51)\ \mu m^2/s$. (c) YG-100 nm beads with $I_{LS} = 80\,W/cm^2$. Result from global fit: $D_{20W,\text{global}} = (3.63 \pm 0.53)\ \mu m^2/s$. The theoretical value of $(3.85 \pm 0.63)\ \mu m^2/s$ [154] is indicated as a dashed black line, its error as a gray area.

E.3. Detailed measurements of various fluorescent dyes in solution

All measurements were performed with the CHSPAD sensor with microlenses. Details on the measurement protocol are given in chapter 6. Section 6.7 summarizes the results.

FluoSpheres YG $\varnothing = 100\,\mathrm{nm}$ fluorescent beads

Figure E.4 shows measurements of different dilutions of *QDot-525* in water. The obtained diffusion coefficient is in good agreement with the theoretical value.

QDot-525 streptavidin ITK

In addition to the measurements presented in section 6.6.1, Figure E.5 shows various measurements of *QDot-525*, with different settings for the binning (1×1, 2×2 and 3×3, left to right). Figure E.6 shows the measured concentration in relation to measurements on a confocal setup of the same samples. The resulting median diffusion coefficients are in good agreement with the theory (sphere with $DW = 15\,\mu\mathrm{m}^2/\mathrm{s} to 20\,\mu\mathrm{m}^2/\mathrm{s}$).

Alexa-488

In addition to the values shown in section 6.6.1, the measured diffusion coefficients of a dilution series of *Alexa-488* are shown in Figure E.7. Figure E.8 shows the corresponding histograms including GAUSSIAN fits. Table E.1 summarizes the measured diffusion coefficients and the particle concentration (from Figure 6.18). The median values, especially for 3×3 binning, are in good agreement with the literature value $D_{20,\mathrm{W}\,\mathit{Alexa\text{-}488},\,\mathrm{lit.}} = 407\,\mu\mathrm{m}^2/\mathrm{s}$ [173].

eGFP oligomers

In addition to the measurements presented in section 6.6.2, Figure E.9 shows various measurements of eGFP-oligomers in solution. The median values including reference values are summarized in Table E.2. To account for systematic errors, also the relative diffusion coefficients between each two species are given. The median values of the measured diffusion coefficients, especially for 3×3 binning, were in very good agreement with the results obtained on the same setup with the EMCCD camera. The overall reduced sensitivity of the CHSPAD led to significantly higher standard deviations (two to ten-fold). Considering relative diffusion coefficients between every two samples, the results were also in accordance to confocal measurements, that were performed in the same lab.

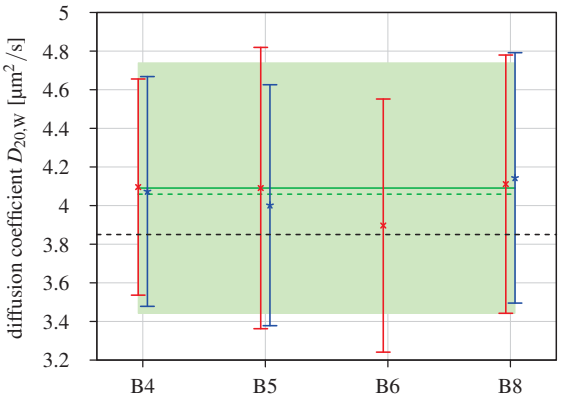


Figure E.4.: Distribution of the diffusion coefficient D of *FluoSpheres YG* $\varnothing = 100\,\text{nm}$ fluorescent beads for different concentrations. A single cross marks a single measurement. Groups of samples reflect multiple measurements. Concentrations ranged from (4.4 ± 0.2) nM down to (0.47 ± 0.05) nM (left to right, obtained with confocal FCS). The theoretical value is shown as a black line, median (dotted,) and mean (solid line, error ranges as light box) values from the measurements are shown in green. Each sample is represented as spot, the theoretical value for a 100 nm sphere, $3.85\,\mu\text{m}^2/\text{s}$, is shown as a dotted black line. A solid green line represents the median, the dotted green line and light green box represent mean value and standard errors. The median value is $D_{20,W} = (4.1 \pm 0.6)\,\mu\text{m}^2/\text{s}$.

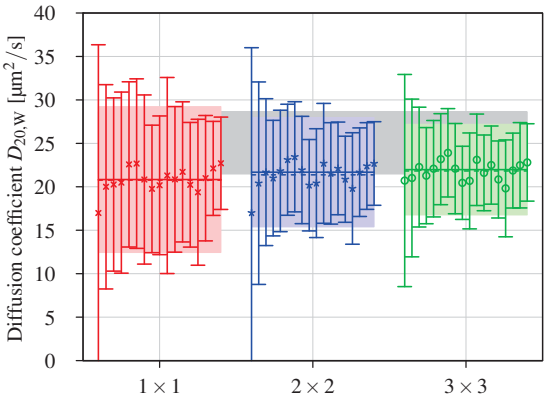


Figure E.5.: Influence of binning on the diffusion coefficient D of *QDot-525 streptavidin ITK*. Three different bin-sized were applied (1×1, 2×2 and 3×3). Results are sorted by laser intensity (0.5 mW to 30 mW). Light gray boxes indicate ranges for a 15 nm to 20 nm sphere. The resulting median diffusion coefficients are $D_{1\times1} = (20.8 \pm 8.4)\,\mu\text{m}^2/\text{s}$, $D_{2\times2} = (21.7 \pm 6.3)\,\mu\text{m}^2/\text{s}$ and $D_{3\times3} = (22.0 \pm 5.2)\,\mu\text{m}^2/\text{s}$.

Sample	$D_{20,W}$ [µm^2/s]	c_{vD}	c [nM]	c_{vc}
A10	426 ± 53	0.12	300 ± 38	0.12
A11	365 ± 87	0.24	180 ± 44	0.24
A12	460 ± 113	0.25	77 ± 20	0.26
A13	388 ± 59	0.15	48.2 ± 6.9	0.14
A14	393 ± 53	0.14	21.8 ± 3.2	0.15
A15	382 ± 55	0.14	13.5 ± 2.4	0.18

Table E.1.: Diffusion coefficients, D, of *Alexa-488* in water for different particle concentrations. Different dilutions were created from the same stock solution. Corresponding distributions of the diffusion coefficients are also shown in Figure E.8 and Figure E.7. Raw data has been binned (3×3) and fitted globally. The literature value is $D_{20,W\textit{Alexa-488},\text{lit.}} = 407\,\mu\text{m}^2/\text{s}$ [173].

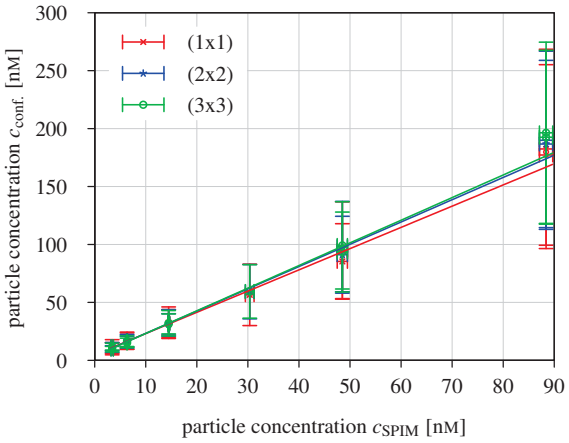


Figure E.6.: Quantum dots: Particle concentration in the focus as measured with the SPIM $c_{\mathbf{SPIM}}$ in relation to results from confocal FCS $c_{\mathbf{conf.}}$. Weighted linear fits per dilution series are shown as line. Mean slope from fit: $s_{1\times1} = (1.83 \pm 0.06)$, $s_{2\times2} = (1.92 \pm 0.07)$ and $s_{3\times3} = (1.95 \pm 0.08)$. For each concentration two datasets were recorded. The errors of the confocal measurements were not taken into account.

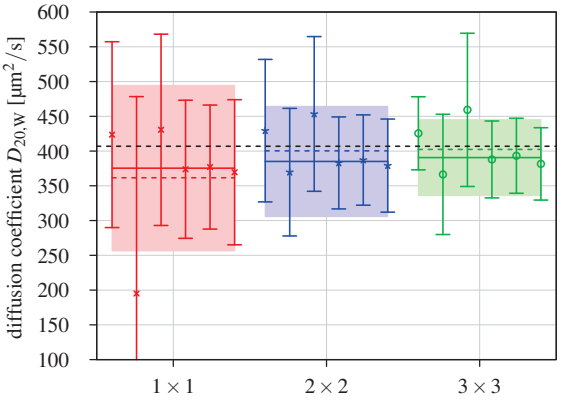


Figure E.7.: Influence of binning on the diffusion coefficient, D, of *Alexa-488* in water. Different bin sized were applied (1×1 to 3×3). The median value is shown for each set as a dashed line, the mean value including standard deviation as a solid line with a surrounding semitransparent area. The absolute values of the 3×3 binning are also shown in Table E.1. In each set, measurements from left to right are A10 to A15. Median values of all six measurements are $D_{1\times1} = (375.4 \pm 119.0)\,\mu m^2/s$, $D_{2\times2} = (385.0 \pm 79.3)\,\mu m^2/s$ and $D_{3\times3} = (390.6 \pm 54.6)\,\mu m^2/s$. The literature value $D_{20,W_{Alexa\text{-}488,\,lit.}} = 407\,\mu m^2/s$ [173] is shown as a black dotted line.

Sample	CHSPAD		EMCCD	FCS
	1×1 binning	3×3 binning		
$D_{20,W_{eGFP\text{-}1x}}$ [µm²/s]	102 ± 68 (0.67)	116 ± 34 (0.30)	118.0 ± 8.0	-
$D_{20,W_{eGFP\text{-}2x}}$ [µm²/s]	71 ± 43 (0.61)	77 ± 21 (0.27)	76.9 ± 8.6	-
$D_{20,W_{eGFP\text{-}3x}}$ [µm²/s]	59 ± 20 (0.34)	61 ± 11 (0.18)	-	-
$D_{20,W_{eGFP\text{-}4x}}$ [µm²/s]	57 ± 18 (0.32)	57 ± 9 (0.16)	62.7 ± 6.0	-
$D_{20,W_{eGFP\text{-}1x}}/D_{20,W_{eGFP\text{-}2x}}$	1.4 ± 1.3	1.5 ± 0.6	1.5 ± 0.2	1.3
$D_{20,W_{eGFP\text{-}1x}}/D_{20,W_{eGFP\text{-}3x}}$	1.7 ± 1.3	1.9 ± 0.7	-	1.6
$D_{20,W_{eGFP\text{-}1x}}/D_{20,W_{eGFP\text{-}4x}}$	1.8 ± 1.3	2.0 ± 0.7	1.9 ± 0.2	2.1
$D_{20,W_{eGFP\text{-}2x}}/D_{20,W_{eGFP\text{-}4x}}$	1.2 ± 0.8	1.4 ± 0.4	1.2 ± 0.2	1.6

Table E.2.: Diffusion coefficients of eGFP oligomers in water determined by different setups. All values are given in $\mu m^2/s$ and are re-calibrated to $\vartheta = 20\,°C$. Data obtained with the CHSPAD stems from single measurements at 19.2 mW (*cf.* Figure E.9). EMCCD data were taken on the same instrument (*cf.* [119]). Confocal FCS data were measured in the same lab and were taken from Ref. [59]. Values in braces give the relative fraction of the standard deviation on the absolute value.

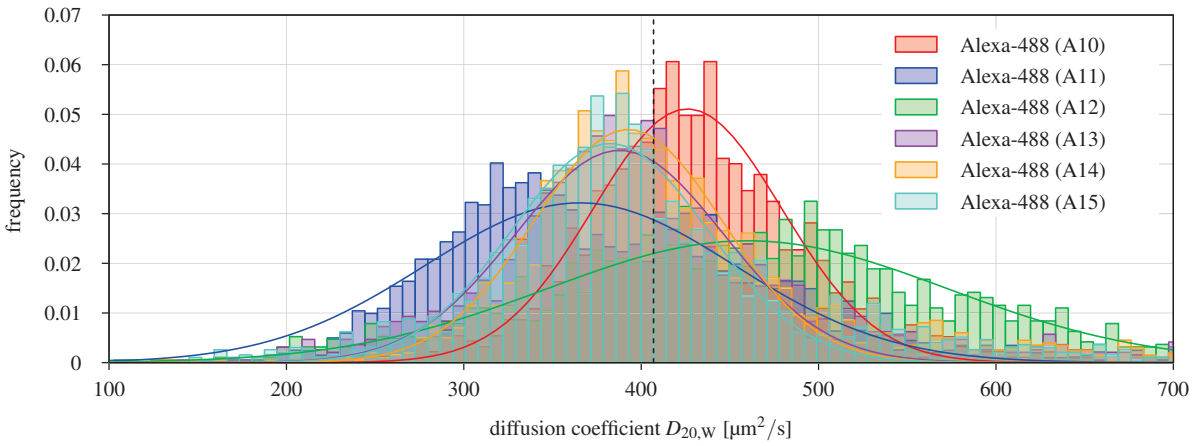


Figure E.8.: ***Alexa-488*** **in solution: Histogram of the the measured diffusion coefficient of different dilutions.** Raw data has been binned (3×3) and fitted globally. A GAUSSIAN fit is applied per histogram. The literature value $D_{20,W_{Alexa\text{-}488}} = 407\,\mu m^2/s$ ([173]) is shown as a black line. Table E.1 shows a summary of the obtained values.

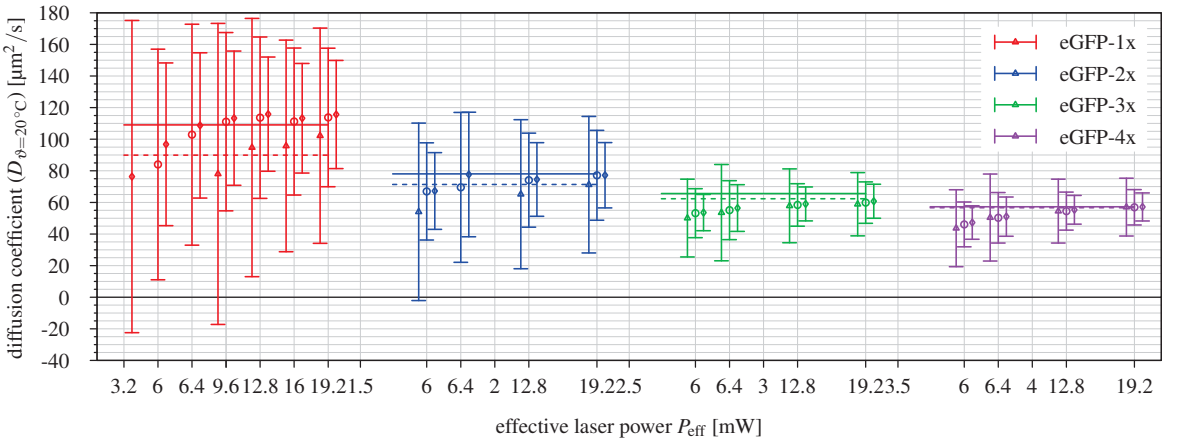


Figure E.9.: Diffusion coefficients of different enhanced green fluorescent protein (eGFP)-oligomers in water determined with the CHSPAD for several laser intensities. Points represent a single measurements. Diffusion coefficients were obtained by fitting a GAUSSIAN to the distribution of diffusion coefficients. Error bars indicate standard deviations. For each laser power a group of three single measurements represents the influence of binning (△:1×1, ○:2×2, ◇:3×3) prior to fitting the correlation estimates. To obtain diffusion coefficients per pixel, raw data was fitted globally. The region of interest (ROI) was 512×4 pixels. Horizontal lines are theoretical values according to Table E.2 either based on a cylinder (solid line) or a sphere (dotted line).

E.4. Performance evaluation of the *RADHARD2* SPAD array

The first sensor that was available an therefore was evaluated in the SPIM-FCS setup first, was the *RADHARD2* SPAD array. This SPAD array has a low fill-factor and thus a low light-collection efficiency. This problem was addressed by not using the standard tube-lens for NIKON microscopes objectives ($f_{\mathrm{TL}} = 200\,\mathrm{mm}$), but a lens with half the focal length. This effectively halves the magnification but increases the light intensity by a factor of four. At the same time this should have only a minor influence on the point spread function (PSF) of each pixel, as the focus size (*i.e.*, the PSF) is mainly determined by the 60× NA 1 detection objective (even at 30× the SPAD size in the image plane stays sub-diffractive).

To test the overall performance of the *RADHARD2* SPAD array, several samples with a biological relevance were tested. The low PDP was further compensated by setting laser power to the maximum $P_{\mathrm{Laser}} = 60\,\mathrm{mW}$ (without additional filters) which resulted in an intensity of $I_{\mathrm{LS}} \simeq 800\,\mathrm{W/cm^2}$ in the center of the lightsheet. Temporal binning (summation of 3 consecutive frames, which reduces the frame time to 3.3 µs) was used to further increase photon sensitivity, as it lowers the probability to miss a photon (see section 4.1.2, with these settings no raw data could be recorded). The raw data was correlated using the FPGA correlator at full speed (10 µs minimum correlation time). Except for the minor differences described above, the same measurement protocol as for the CHSPAD was used (see section 6.1).

Figure E.10 shows the mean ACCs red, averaged over all pixels and an ACC from a representative single pixel blue for four different samples, *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads (*T7279*, INVITROGEN) (a), *QDot-525 streptavidin ITK* (*Q10041MP*, INVITROGEN) (b), *Alexa-488* (INVITROGEN) fluorescent dye (d), and eGFP tetramers (c). The three-dimensional diffusion model including afterpulsing and offset correction was used for the fits (see section 2.4.3 and section 2.4.3 for the models). Model functions without afterpulsing are shown as dashed thick lines. In case of the *TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads, the measurement duration was 42 s. The duration of all other measurements was approximately 165 s. Sample temperature was 24 °C in all cases.

As the raw data could not be recorded in parallel, no bleach correction was applied. Thus, the significant bleaching in case of the eGFP tetramers (c) was accounted for by a two-component fit model of the ACCs (sum of two three-dimensional diffusion models).

Figure E.11 shows the distribution of the diffusion coefficients obtained by fits of the ACCs for the same samples. The ACCs of the single pixels of all samples showed a significant amount of noise. This resulted in a broad distribution of diffusion coefficients as depicted in Figure E.11. Only the two brightest samples (*TetraSpeck* $\varnothing = 100\,\mathrm{nm}$ fluorescent beads and *QDot-525 streptavidin ITK*) showed a GAUSSIAN shape of the histogram which allowed to extract a mean value. For the two other sample, a single GAUSSIAN curve did not match the distribution and a sum of two curves was used for model fitting. Table E.3 summarizes the results of the measurements. In addition to the mean diffusion coefficients ob-

	measurement			theory
sample	$D_{20,\mathrm{W}}$ [µm²/s]	$c_{\mathrm{v}\,D_{20,\mathrm{W}}}$	CPM [Hz]	$D_{20,\mathrm{W}}$ [µm²/s]
TetraSpeck beads	3.0 ± 0.7	0.23	349	~ 4
QDot-525	16 ± 6	0.37	50	~ 25
eGFP tetramers	55[1]	-	-	~ 60
Alexa-488	75[1]	-	32	~ 410

[1] The fit of the histogram was done using the sum of two GAUSSIAN functions. Thus, the standard deviation is not meaningful.

Table E.3.: Diffusion coefficients of various samples measured with the *RADHARD2* SPAD array. Shown results are mean values and standard deviations obtained from GAUSSIAN fits of the histogrammed shown in Figure E.11. For theoretical values and details on the dyes see section 2.2.4. The third column shows the photon count rate divided by the number of particles obtained from fit, the so called count rate per molecule (CPM). In case of eGFP tetramers, no value could be extracted as the bleaching has a strong effect on the particle number.

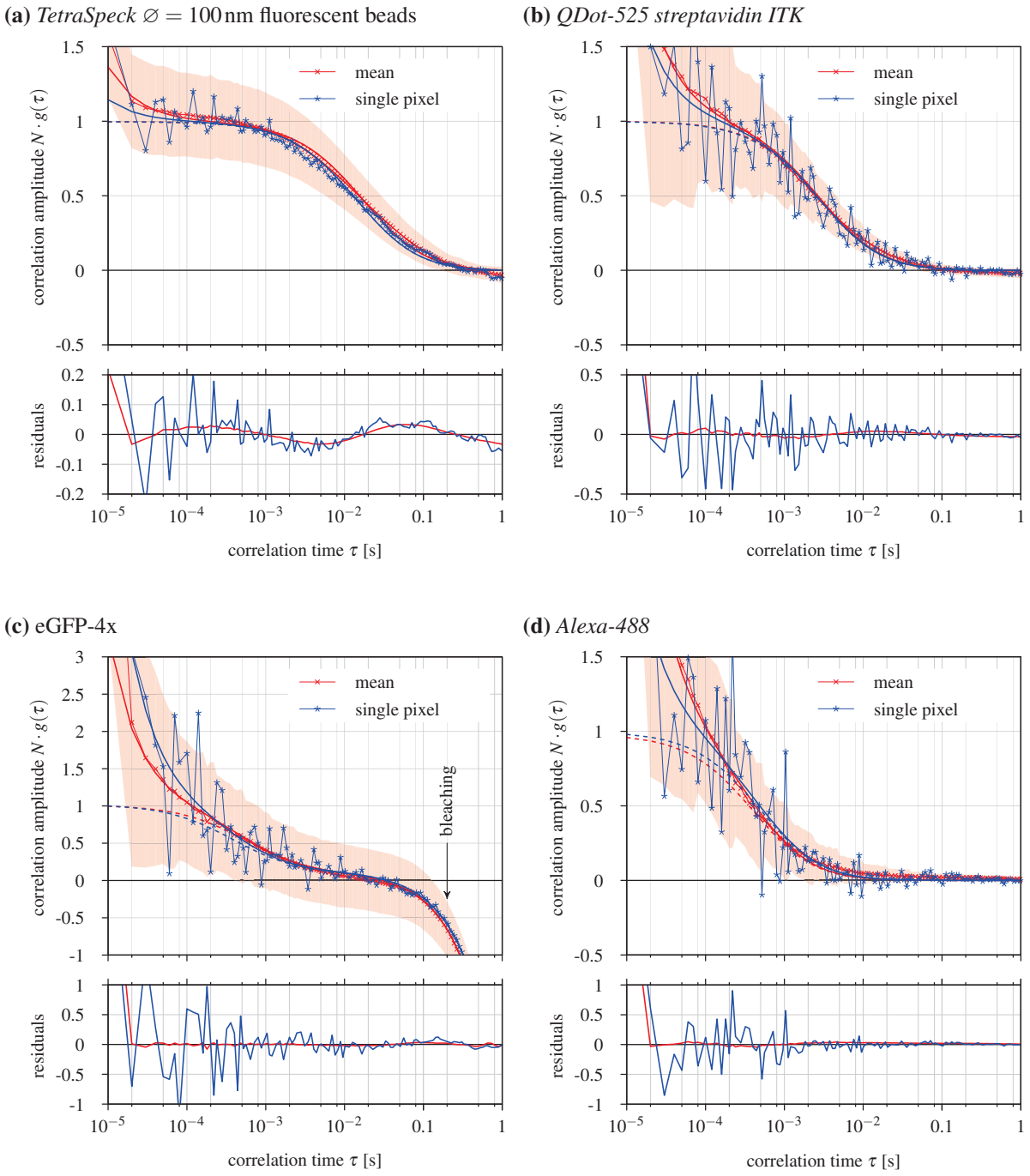


Figure E.10.: Example measurement of various samples in water using the *RADHARD2* SPAD array. The mean ACCs over all pixels are shown in red with the standard deviation as a light red area. An example curve of a single pixel is shown in blue. The fit of the full model function is shown as a solid line (three-dimensional diffusion model including afterpulsing and offset-correction), the model function without afterpulsing is plotted as a dashed graph. Residuals are shown below of each plot. (a) *TetraSpeck* ∅ = 100 nm fluorescent beads. (b) *QDot-525 streptavidin ITK*. (c) eGFP tetramers. (d) *Alexa-488* fluorescent dye. All measurements were done at maximum laser intensity $I_{LS} = 800\,W/cm^2$, at $\vartheta = 24\,°C$. In (b), (c) and (d) afterpulsing is taken into account. Raw data was not bleach-corrected, thus the effect of bleaching is clearly visible as a strong decay around $\tau = 0.1\,s$ in (c). This was accounted for by a two-component fit (sum of two three-dimensional diffusion models). Defective pixels were excluded.

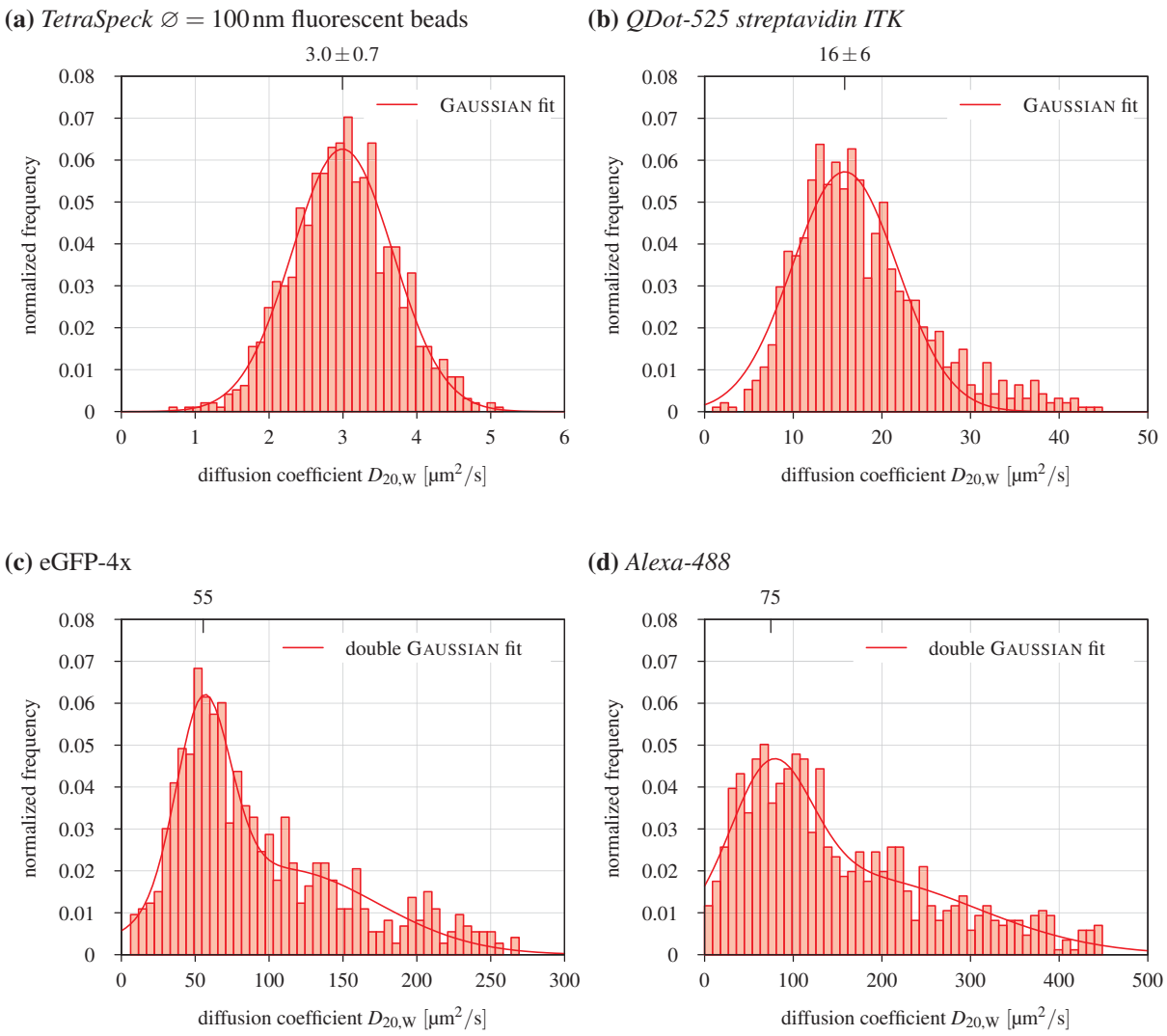


Figure E.11.: Histogram of the diffusion coefficients of all pixels of the *RADHARD2* SPAD array for various samples. The mean values from the fits are shown above the plots. (a) *TetraSpeck* ∅ = 100 nm fluorescent beads. (b) *QDot-525 streptavidin ITK*. (c) eGFP tetramers. (d) *Alexa-488* fluorescent dye. Focal parameters were $w_{xy} = (490 \pm 70)$ nm and $w_z = (1105 \pm 310)$ nm (obtained from bead-scan).

tained from GAUSSIAN fits of the histograms, the mean count rate per molecule during the measurement is given. Simulations showed (*cf.*, section 6.4), that a value above 300 Hz per molecule is required for an accurate determination of the diffusion coefficient (*i.e.*, relative standard deviation (rel. sd.) < 0.2).

E.5. Conclusion

To measure the diffusion coefficient with high precision, a certain number of photon counts per molecule have to be detected. These measurements showed that the PDP of the *RADHARD2* is too low for samples less bright than *TetraSpeck* $\varnothing = 100\,\text{nm}$ fluorescent beads. Therefore, measurements with biologically relevant dyes were not feasible. By pushing incident light intensity to the max (more than ten fold of what is typically used for *in-vivo* measurements), it was at least possible to measure mean diffusion coefficients of *QDot-525 streptavidin ITK* and eGFP tetramers that were both in range of the theoretical value.

Regarding the afterpulsing, the *RADHARD2* showed a similar behavior as the CHSPAD. The afterpulsing component was clearly visible in all ACCs and its amplitude increases with the decreasing fluorescence intensity. In case of *Alexa-488*, the afterpulsing interfered with the diffusion time and inhibited proper fitting. Together with the low PDP, this lead to a four-fold underestimation of the diffusion coefficient.

To reduce the number of missed photons at the high fluorescence intensity, the *RADHARD2* was operated at maximum speed. At these settings, a parallel acquisition of the raw data was not possible. Therefore, no bleach-correction could be performed. Due to the lack of support for cross-correlation in the FPGA-based correlator, global fits could not be carried out. The overall poor PDP and the availability of the CHSPAD with microlenses led to a discontinuation of the enhancement and development of the *RADHARD2* platform.

E.6. Recalibration

To account for different sample temperatures, diffusion coefficients need to be re-calibrated to 20 °C to allow comparison. According to the STOKES-EINSTEIN equation (equation (1.4)), a diffusion coefficient D_0 at temperature T_0 and viscosity η_0 can be re-calibrated to conditions T_1 and η_1 as follows:

$$D_1 = \frac{D_0 \cdot T_1}{T_0} \cdot \frac{\eta_0}{\eta_1}, \tag{E.1}$$

Assuming water as the solvent, a good estimation for the viscosity is [68]:

$$\eta_\text{W}(T) = A \cdot 10^{B/(T-C)}, \quad \text{with } A = 2.414 \times 10^{-5}\,\text{Ns/m}^2, B = 247.8\,\text{K}, C = 140\,\text{K}. \tag{E.2}$$

Diffusion coefficients given in this thesis are re-calibrated to $T = 20\,°\text{C}$, so called $D_{20,\text{W}}$.

E.7. Image sharpness

For aligning the instrument, 100 nm fluorescent beads are used. Due to their size and speed those beads are good distinguishable when being imaged with the EMCCD camera using standard settings. The CHSPAD can also be used for focusing, but the lower quantum efficiency and thus a higher integration time makes it difficult to distinguish between single beads (smearing). To have an objective measure for the sharpness of the live-image, a sharpness-factor s is introduced. Under the assumption that a sharper image has more prominent edges, an edge detection is performed and the resulting gray-scale values are added up [161]. For edge detection, a discrete Laplace operator $\mathbf{D}^2_{xy}$is used:

$$s = \sum_{xy} \left(\mathbf{D}^2_{xy} * I_{xy} \right) \tag{E.3}$$

with

$$\mathbf{D}^2_{xy} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{E.4}$$

Nomenclature

$\varepsilon_{\text{fluor}}$	Molar extinction coefficient of a fluorophore.
η_{det}	Quantum efficiency.
SNR	Signal to noise ratio.
Var	Variance
ϕ_{fluor}	Quantum yield of a fluorophore.
τ_{min}	Minimum correlation time.
τ_D	Diffusion time.
τ_k	Correlation time of lag k.
$\text{CF}_{\mu\text{L}}$	Concentration factor of the microlenses.
FF	Fill-factor of a sensor.
MDE	molecular detection efficiency.
ϑ	Temperature in °C.
B_{fluor}	Brightness of a fluorophore.
c	Particle concentration in the focus.
$D_{20,\text{W}}$	Diffusion coefficient corrected for standard conditions ($\vartheta = 20°$, viscosity of water).
D	Diffusion coefficient.
f_{TL}	Focal length of the tube-lens.
G_k	Not normalized correlation function estimate at position k.
g_k	Normalized correlation function estimate at position k.
G	Correlation function, used as model.
g	Correlation curve obtained from correlator.
I^d	Delayed intensity signal in a correlator.
I^u	Undelayed intensity signal in a correlator.
I_{LS}	Laser intensity at the center of the lightsheet.
I_k	Raw intensity value at position k.
m	Binning ratio of a multiple-τ correlator. Typically, $m = 2$.

N_{threads} Number of threads.

N_{VU} Amount of vertical-unrolling.

N Particle number in the focus.

P_{eff} Effective laser power.

R_{h} Hydrodynamic radius.

s_c Chunk size. The number of input values processed by a correlator in a sequence before a context switch is performed.

V_{SPAD} Voltage of the SPAD.

V_{bias} Voltage bias of the SPAD.

V_{break} SPAD breakdown voltage.

V_{DD} Positive logic supply voltage.

V_{eff} Effective observation volume.

V_{ex} Excess voltage above breakdown of the SPAD.

V_{obs} Size of the observation volume.

V_{OP} SPAD operating voltage.

V_{Q} Supply voltage of the quenching transistor.

V_{Top} Tunable SRAM resistors of CHSPAD.

w_{obs} Diameter of the sphere forming the observation volume.

w_{xy} Lateral half axis of the focus GAUSSIAN.

w_z Longitudinal half axis of the focus GAUSSIAN.

c_{v} Relative standard deviation (rel. sd.).

Acronyms

ACC autocorrelation curve.
ACF autocorrelation function.
ADC analogue-to-digital converter.
ADU analog-to-digital unit.
AGU address generation unit.
ALU arithmetic logic unit.
APD avalanche photodiode.
API application programming interface.
APS active pixel sensor.
ARM advanced RISC machines.
ASIC application-specific integrated circuit.
AVX advanced vector extensions.

BIOS basic input output system.
BM BROWNIAN motion.
BRAM block RAM.

CCC cross-correlation curve.
CCD charge coupled device.
CCF cross-correlation function.
CE correlation estimate.
CHSPAD Swiss single photon avalanche diode array.
CLB configurable logic block.
CMOS complementary metal-oxide-semiconductor.
CPM count rate per molecule.
CPU central processing unit.
CRC cyclic redundancy check.
CUDA compute unified device architecture.

DC direct current.
DCM digital clock manager.
DCR dark count rate.
DDR double data rate.
DFE dataflow engine.
DKFZ German cancer research center.
DNA deoxyribonucleic acid.
DOE diffractive optical element.
DP double precision.
DSP digital signal processor.

ECC error-correcting code.
eGFP enhanced green fluorescent protein.
EM electron multiplication.
EMCCD electron multiplying charge coupled device.
EPFL École polytechnique fédérale de Lausanne.

FCCS fluorescence cross-correlation spectroscopy.
FCS fluorescence correlation spectroscopy.
FEP fluorinated ethylene propylene.
FF flip-flop.
FFT fast Fourier transform.
FIFO first in - first out buffer.
FLOP floating-point operation.
FLOPS floating-point operation per second.
FMA fused multiply accumulate.
FOV field of view.
FP floating-point.
FPGA field programmable gate array.
FPU floating-point unit.
FRAP fluorescence recovery after photobleaching.

GDDR graphics double data rate.
GFP green fluorescent protein.
GMM gimbal mounted mirror.
GPGPU general purpose computation on graphics processing unit.
GPS global positioning system.
GPU graphics processing unit.
GUI graphical user interface.

HDD hard disk drive.
HDL hardware description language.
HeLa Henrietta Lacks.
HPC high pin-count connector.
HPC high performance computing.
HT hyper-threading.

IC integrated circuit.
IEEE Institute of Electrical and Electronics Engineers.
imaging FCS imaging fluorescence correlation spectroscopy.

IO input/output.
IOB input/output block.

LCD liquid crystal display.
LED light emitting diode.
LFSR linear feedback shift register.
LSFM lightsheet fluorescence microscopy.
LSM lightsheet microscopy.
LUT lookup table.

MAC multiply accumulate.
MDE molecular detection efficiency.
MIMD multiple instruction, multiple data.
MIS metal-insulator-semiconductor.
MISD multiple instruction, single data.
MMX multimedia extension.
mRFP monomeric red fluorescent protein.
MSD mean squared displacement.
MUX multiplexer.

NA numerical aperture.
ND neutral density.
NMOS N-type metal-oxide-semiconductor.

OD optical density.
OpenCL open computing language.
OS operating system.

PC personal computer.
PCB printed circuit board.
PCIe peripheral component interconnect express.
PD photodiode.
PDP photon detection probability.
PLL phase-locked loop.
PM photon-mode.
PMT photomultiplier tube.
PowerPC performance optimization with enhanced RISC performance chip.
PSF point spread function.

QD quantum dot.
QE quantum efficiency.

RADAR radio detection and ranging.
RAID redundant array of inexpensive disks.
RAM random access memory.
rel. sd. relative standard deviation.
RISC reduced instruction set computer.
RMS root mean square.
ROI region of interest.
RTL register transfer level.

SATA serial advanced technology attachment.
sCMOS scientific CMOS.
sd standard deviation.
SDK software development kit.
SDR single data rate.
SDRAM synchronous dynamic random access memory.
SIMD single instruction, multiple data.
SIMT single instruction, multiple thread.
SISD single instruction, single data.
SLM spatial light modulator.
SMX streaming multiprocessor.
SNR signal to noise ratio.
SP single precision.
SPAD single photon avalanche diode.
SPIM selective plane illumination microscopy.
SPT single particle tracking.
SRAM static random access memory.
SSD solid state disk.
SSE soft streaming extension.

TCP/IP transmission control protocol / internet protocol.
TCSPC time-correlated single-photon counting.
TDC time to digital converter.
TIRF total internal reflection.
TIRFCS total internal reflection fluorescence correlation spectroscopy.
TUPC time-uncorrelated photon-counting.

USB universal serial bus.

VHDL Very High Speed Integrated Circuit Hardware Description Language.

YG yellow-green.

List of Figures

List of Tables

List of Listings

List of Algorithms

Bibliography

[1] Evga - product specs - evga geforce gtx 780 ti. https://www.evga.com/Products/Specs/GPU.aspx?pn=DA9CFD1C-4A26-4DFE-8EA4-60BFEE692152. accessed 2015-02-09.

[2] Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70, Aug 2008. doi: 10.1109/IEEESTD.2008.4610935.

[3] Maxeler technologies products. http://www.maxeler.com/products/, March 2014. accessed 2014-03-17.

[4] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland Science (Taylor & Francis Group), 2002.

[5] ALV-Laser Vertriebsgesellschaft.m.b.H. Alv-products : Digital correlator. http://www.alvgmbh.de/Products/Correlators/correlators.html, 2012. accessed 2015-04-25.

[6] AMD, Inc. HBM. http://www.amd.com/de-de/innovations/software-technologies/hbm. accessed 2015-08-19.

[7] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM, 1967.

[8] Andor. *Andor sCMOS Brochure*, 2014. URL http://www.andor.com/pdfs/literature/Andor_sCMOS_Brochure.pdf.

[9] Andor. *Andor Clara Series Specifications*, 2014. URL http://www.andor.com/pdfs/specifications/Andor_Clara_Series_Specifications.pdf.

[10] Andor. *Andor iXon Ultra and iXon 3 EMCCD Brochure*, 2014. URL http://www.andor.com/pdfs/literature/Andor_iXon3_EMCCD_Brochure.pdf.

[11] Andor Technology Ltd. Andor iXon3 specifications. https://www.andor.com/pdfs/specifications/Andor_iXon_860_Specifications.pdf, April 2015.

[12] D Axelrod, D E Koppel, J Schlessinger, E Elson, and W W Webb. Mobility measurement by analysis of fluorescence photobleaching recovery kinetics. *Biophysical journal*, 16(9):1055–69, 1976. doi: 10.1016/S0006-3495(76)85755-4.

[13] Kirsten Bacia, Sally A. Kim, and Petra Schwille. Fluorescence cross-correlation spectroscopy in living cells. *Nature Methods*, 3(2):83–89, 2006.

[14] David F Bacon, Rodric Rabbah, and Sunil Shukla. Fpga programming for the masses. *Communications of the ACM*, 56(4):56–63, 2013.

[15] Nirmalya Bag and Thorsten Wohland. Imaging fluorescence fluctuation spectroscopy: New tools for quantitative bioimaging. *Annual Review of Physical Chemistry*, 65(1):131213171031009, 2012. doi: 10.1146/annurev-physchem-040513-103641.

[16] Nirmalya Bag, Jagadish Sankaran, Alexandra Paul, Rachel S. Kraut, and Thorsten Wohland. Calibration and limits of Camera-Based fluorescence correlation spectroscopy: A supported lipid bilayer study. *ChemPhysChem*, 13(11):2784–2794, 2012. doi: 10.1002/cphc.201200032.

[17] Nirmalya Bag, Ashraf Ali, Virander Singh Chauhan, Thorsten Wohland, and Aseem Mishra. Membrane destabilization by monomeric hIAPP observed by imaging fluorescence correlation spectroscopy. *Chemical Communications*, 49(80):9155, 2013. doi: 10.1039/C3CC44880K.

[18] Nirmalya Bag, Darilyn Hui Xin Yap, and Thorsten Wohland. Temperature dependence of diffusion in model and live cell membranes characterized by imaging fluorescence correlation spectroscopy. *Biochimica et Biophysica Acta (BBA) - Biomembranes*, 1838(3):802–813, 2014. doi: 10.1016/j.bbamem.2013.10.009.

[19] Nina Baudendistel, Gabriele Müller, Waldemar Waldeck, Peter Angel, and Jörg Langowski. Two-hybrid fluorescence cross-correlation spectroscopy detects protein-protein interactions in vivo. *Chemphyschem : a European journal of chemical physics and physical chemistry*, 6(5):984–90, 2005. doi: 10.1002/cphc.200400639.

[20] S. Bellisai, L. Ferretti, F. Villa, A. Ruggeri, S. Tisa, A. Tosi, and F. Zappa. Low-power 20-meter 3d ranging SPAD camera based on continuous-wave indirect time-of-flight. *Proc. of SPIE*, 8375: 83750E–83750E–7, 2012. doi: 10.1117/12.920407.

[21] K.M. Berland, P.T. So, and E. Gratton. Two-photon fluorescence correlation spectroscopy: method and application to the intracellular environment. *Biophysical Journal*, 68(2):694–701, 1995. doi: 10.1016/S0006-3495(95)80230-4.

[22] Felix Bestvater, Zahir Seghiri, Moon Sik Kang, Nadine Gröner, Ji Young Lee, Im Kang-Bin, and Malte Wachsmuth. EMCCD-based spectrally resolved fluorescence correlation spectroscopy. *OPTICS EXPRESS*, 18(23):23818–23828, 2010.

[23] Christophe Bobda. *Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications*. Springer Science & Business Media, 2007.

[24] Daniel Boening, Teja W. Groemer, and Jürgen Klingauf. Applicability of an EM-CCD for spatially resolved TIR-ICS. *OPTICS EXPRESS*, 18(13):13516–13528, 2010.

[25] P. Brazda, J. Krieger, B. Daniel, D. Jonas, T. Szekeres, J. Langowski, K. Toth, L. Nagy, and G. Vamosi. Ligand binding shifts highly mobile retinoid x receptor to the Chromatin-Bound state in a Coactivator-Dependent manner, as revealed by Single-Cell imaging. *Molecular and Cellular Biology*, 34(7):1234–1245, 2014. doi: 10.1128/MCB.01097-13.

[26] Peter Brazda. *Determination of dynamic properties of nuclear receptors*. PhD thesis, 2014. URL `http://hdl.handle.net/2437/179680`.

[27] Michael Brinkmeier, Klaus Dörre, Jens Stephan, and Manfred Eigen. Two-beam cross-correlation: a method to characterize transport phenomena in micrometer-sized structures. *Analytical chemistry*, 71(3):609–616, 1999.

[28] Robert Brown. A brief account of microscopical observations made in the months of june, july and august 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies. *The Philosophical Magazine and Annals of Philosophy*, 4(21):161–173, 1828.

[29] Samuel Burri. tba. 2015.

[30] Samuel Burri, Damien Stucki, Yuki Maruyama, Claudio Bruschini, Edoardo Charbon, and Francesco Regazzoni. Jailbreak imagers: Transforming a Single-Photon image sensor into a true random number generator. In *International Image Sensor Workshop*, 2013.

[31] Samuel Burri, D Stucky, Yuki Maruyama, Claudio Bruschini, Edoardo Charbon, and Francesco Regazzoni. Jailbreak imagers: transforming a single-photon image sensor into a true random number generator. 2013.

[32] Samuel Burri, Yuki Maruyama, Xavier Michalet, Francesco Regazzoni, Claudio Bruschini, and Edoardo Charbon. Architecture and applications of a High
Resolution, gated SPAD image sensor. *Optics Express*, submitted, 2014.

[33] Samuel Burri, Yuki Maruyama, Xavier Michalet, Francesco Regazzoni, Claudio Bruschini, and Edoardo Charbon. Architecture and applications of a high resolution gated spad image sensor. *Optics express*, 22(14):17573–17589, 2014.

[34] J. Capoulade, M. Wachsmuth, L. Hufnagel, and M. Knop. Quantitative fluorescence imaging of protein diffusion and interaction in living cells. *Nat. Biotechnol.*, 2011. doi: 10.1038/nbt.1928.

[35] L. Carrara, C. Niclass, N. Scheidegger, H. Shea, and E. Charbon. A gamma, x-ray and high energy proton radiationtolerant cmos image sensor for space applications. In *IEEE International Solid-State Circuits Conference*, pages 40–41. Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International, 2009. doi: 10.1109/ISSCC.2009.4977297.

[36] Lucio Carrara, Marek Gersbach, and Edoardo Charbon. Low cost earth sensor based on oxygen airglow: D1 - spad sensor design report, December 2007.

[37] Riccardo Cattaneo, Christian Pilato, Matteo Mastinu, Oliver Kadlcek, Oliver Pell, and Marco Domenico Santambrogio. Runtime adaptation on dataflow hpc platforms. In *Adaptive Hardware and Systems (AHS), 2013 NASA/ESA Conference on*, pages 84–91. IEEE, 2013.

[38] Martin Chalfie, Yuan Tu, Ghia Euskirchen, William W Ward, and Douglas C Prasher. Green fluorescent protein as a marker for gene expression. *Science*, 263(5148):802–805, 1994.

[39] C Coates, Boyd Fowler, and Gerhard Holst. sCMOS: scientific CMOS technology. Technical report, 2009. URL `http://www.andor.com/pdfs/learning/technical/neo/NewsCMOSvs.CurrentMicroscopyCameras.pdf`.

[40] R.A. Colyer, G. Scalia, T. Kim, I. Rech, D. Resnati, S. Marangoni, M. Ghioni, S. Cova, S. Weiss, and X. Michalet. High-throughput multispot single-molecule spectroscopy. In *Proceedings-Society of Photo-Optical Instrumentation Engineers*, volume 7571, page 75710G. NIH Public Access, 2010. doi: 10.1117/12.841398.

[41] R.A. Colyer, G. Scalia, F.A. Villa, F. Guerrieri, S. Tisac, F. Zappa, S. Covab, X. Weiss S., and Michalet. Ultra high-throughput single molecule spectroscopy with a 1024 pixel SPAD. *Proc. of SPIE*, 7905:790503—-1, 2011.

[42] Intel Corporation. Intel intrinsics guide. https://software.intel.com/sites/landingpage/IntrinsicsGuide/, . accessed 2015-05-29.

[43] Intel Corporation. Intel core i7-4770 processor. http://ark.intel.com/de/products/75122, . accessed 2015-05-29.

[44] Intel Corporation. Intel core i7-5960x processor extreme edition. http://ark.intel.com/de/products/82930, . accessed 2015-05-29.

[45] Intel Corporation. Intel core i7-4770 processor specifications. http://ark.intel.com/products/75122, March 2014. accessed 2014-03-17.

[46] Correlator.com. Correlator.com : Products. http://correlator.com, 2015. accessed 2015-04-26.

[47] S. Cova, M. Ghioni, A. Lacaita, C. Samori, and F. Zappa. Avalanche photodiodes and quenching circuits for single-photon detection. *Applied Optics*, 35(12):1956, 1996. doi: 10.1364/AO.35.001956.

[48] Michael J. Culbertson, Joshua T. B. Williams, Wayland W. L. Cheng, Dee Ann Stults, Emily R. Wiebracht, John J. Kasianowicz, and Daniel L. Burden. Numerical fluorescence correlation spectroscopy for the analysis of molecular dynamics under nonstandard conditions. *Analytical Chemistry*, 79(11):4031–4039, 2007. doi: 10.1021/ac062013m.

[49] M.J. Culbertson and D.L. Burden. A distributed algorithm for multi-tau autocorrelation. *Rev. Sci. Inst.*, 78(4):044102–044102, 2007.

[50] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55, 1998.

[51] Maxine Dahan, Sabine Lévi, Camille Luccardini, Philippe Rostaing, Béatrice Riveau, and Antoine Triller. Diffusion dynamics of glycine receptors revealed by Single-Quantum dot tracking. *Science*, 302:442–444, 2003.

[52] Williams Davis, Swift Ball, and Matayoshi. Data reduction methods for application of fluorescence correlation spectroscopy to pharmaceutical drug discovery. *Current pharmaceutical biotechnology*, 4(6):451–62, 2003.

[53] Michelle A. Digman, Claire M. Brown, Parijat Sengupta, Paul W. Wiseman, Alan R. Horwitz, and Enrico Gratton. Measuring fast dynamics in solutions and cells with a laser scanning microscope. *Biophysical Journal*, 89(2):1317–1327, 2005. doi: 10.1529/biophysj.105.062836.

[54] Michelle A. Digman, Parijat Sengupta, Paul W. Wiseman, Claire M. Brown, Alan R. Horwitz, and Enrico Gratton. Fluctuation correlation spectroscopy with Laser-Scanning microscope: Exploiting the hidden time structure. *Biophysical Journal*, 88(5):L33, 2005. doi: 10.1529/biophysj.105.061788.

[55] R Dimond, MJ Flynn, O Mencer, and O Pell. Maxware: acceleration in hpc. *IEEE HOT CHIPS*, 20, 2008.

[56] J Dongarra. Luszczek and a. petitet (2001):" the linpack benchmark: Past, present and future", university of tennessee. Technical report, mimeo.

[57] N. Dross, C. Spriet, M. Zwerger, G. Müller, W. Waldeck, and J. Langowski. Mapping eGFP oligomer mobility in living cell nuclei. *PLoS ONE*, 4(4):e5041, 2009. doi: 10.1371/journal.pone.0005041.

[58] Nicolas Dross. *Mobilität von eGFP-Oligomeren in lebenden Zellkernen*. Doktorarbeit, 2009. URL `http://archiv.ub.uni-heidelberg.de/volltextserver/volltexte/2010/10470/pdf/Dross_Mobilitaet_von_eGFP_Oligomeren_in_lebenden_Zellkernen.pdf`.

[59] Nicolas Dross, Corentin Spriet, Monika Zwerger, Gabriele Müller, Waldemar Waldeck, and Jörg Langowski. Mapping egfp oligomer mobility in living cell nuclei. *PLoS One*, 4(4):e5041, 2009.

[60] A. Einstein. über die von der molekularkinetischen Theorie der Wärme geforderte bewegung von in ruhenden Flüssigkeiten suspendierten teilchen. *Annalen der Physik*, 322(8):549–560, 1905. doi: 10.1002/andp.19053220806.

[61] Jörg Enderlein and Ingo Gregor. Using fluorescence lifetime for discriminating detector afterpulsing in fluorescence-correlation spectroscopy. *Review of Scientific Instruments*, 76:033102, 2005. doi: 10.1063/1.1863399.

[62] M. Engels, B. Hoppe, H. Meuth, and R. Peters. A single chip 200 MHz digital correlation system for laser spectroscopy with 512 correlation channels. In *ISCAS'99. Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, 1999*, volume 5, pages 160–163. IEEE, 1999.

[63] M. Engels, B. Hoppe, H. Meuth, and R. Peters. Fast digital photon correlation system with high dynamic range. In *Proceedings of the 13th Annual IEEE International ASIC/SOC Conference, 2000*, pages 18–22. IEEE, 2000.

[64] Harold P Erickson. Size and shape of protein molecules at the nanometer level determined by sedimentation, gel filtration, and electron microscopy. *Biol Proced Online*, 11(1):32–51, 2009.

[65] Adolf Fick. Ueber diffusion. *Annalen der Physik*, 170(1):59–86, 1855.

[66] Michael Flynn, R Dimond, Oskar Mencer, and Oliver Pell. Finding speedup in parallel processors. In *Parallel and Distributed Computing, 2008. ISPDC'08. International Symposium on*, pages 3–7. IEEE, 2008.

[67] Michael J Flynn and Kevin W Rudd. Parallel architectures. *ACM Computing Surveys (CSUR)*, 28 (1):67–70, 1996.

[68] Robert W Fox, Alan T McDonald, and Philip J Pritchard. Introduction to fluid mechanics. 2011.

[69] M. Gersbach, R. Trimananda, Y. Maruyama, M. Fishburn, D. Stoppa, J. Richardson, R. Walker, R. K. Henderson, and E. Charbon. High frame-rate TCSPC-FLIM using a novel SPAD-based image sensor. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7780, page 38, 2010.

[70] Gerard Giraud, Holger Schulze, Day-Uei Li, Till T. Bachmann, Jason Crain, David Tyndall, Justin Richardson, Richard Walker, David Stoppa, Edoardo Charbon, Robert Henderson, and Jochen Arlt. Fluorescence lifetime biosensing with DNA microarrays and a CMOS-SPAD imager. *Biomedical Optics Express*, 1(5):1302, 2010. doi: 10.1364/BOE.1.001302.

[71] Heike Glauner, Ivo R Ruttekolk, Kerrin Hansen, Ben Steemers, Yi-Da Chung, Frank Becker, Stefan Hannus, and Roland Brock. Simultaneous detection of intracellular target and off-target binding of small molecule cancer drugs at nanomolar concentrations. *British Journal of Pharmacology*, 160(4):958–970, 2010. doi: 10.1111/j.1476-5381.2010.00732.x.

[72] S Gong, I Labanca, I Rech, and M Ghioni. A 32-channel photon counting module with embedded auto/cross-correlators for real-time parallel fluorescence correlation spectroscopy. *Review of Scientific Instruments*, 85(10):103101, 2014.

[73] Sixia Gong, Ivan Labanca, Ivan Rech, and Massimo Ghioni. A simple and flexible fpga based autocorrelator for afterpulse characterization of single-photon detectors. In *Multi-Conference on Systems, Signals & Devices (SSD), 2014 11th International*, pages 1–6. IEEE, 2014.

[74] Michael Gösch, Alexandre Serov, Tiemo Anhut, Theo Lasser, Alexis Rochas, Pierre-André Besse, Radivoje S. Popovic, Hans Blom, and Rudolf Rigler. Parallel single molecule detection with a fully integrated single-photon 2x2 CMOS detector array. *Journal of Biomedical Optics*, 9(5):913, 2004. doi: 10.1117/1.1781668.

[75] K. Greger, J. Swoger, and E. H. K. Stelzer. Basic building units and properties of a fluorescence single plane illumination microscope. *Review of Scientific Instruments*, 78:023705, 2007.

[76] Frederik Grüll. *Acceleration of biomedical image processing and reconstruction with FPGAs*. PhD thesis, Univ.-Bibliothek Frankfurt am Main, 2015.

[77] Frederik Grüll, Manfred Kirchgessner, Rainer Kaufmann, Michael Hausmann, and Udo Kebschull. Accelerating image analysis for localization microscopy with fpgas. In *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, pages 1–5. IEEE, 2011.

[78] Syuan-Ming Guo, Nirmalya Bag, Aseem Mishra, Thorsten Wohland, and Mark Bathe. Bayesian total internal reflection fluorescence correlation spectroscopy reveals hIAPP-Induced plasma membrane domain organization in live cells. *Biophysical Journal*, 106(1):190–200, 2014. doi: 10.1016/j.bpj.2013.11.4458.

[79] John L Gustafson. Reevaluating amdahl's law. *Communications of the ACM*, 31(5):532–533, 1988.

[80] Richard W Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

[81] Stefan Harfst, Alessia Gualandris, David Merritt, Rainer Spurzem, Simon Portegies Zwart, and Peter Berczik. Performance analysis of direct n-body algorithms on special-purpose supercomputers. *New Astronomy*, 12(5):357–377, 2007.

[82] Ulrich Haupts, Sudipta Maiti, Petra Schwille, and Watt W. Webb. Dynamics of fluorescence fluctuations in green fluorescent protein observed by fluorescence correlation spectroscopy. *PNAS*, 95:13573–13578, 1998.

[83] Elke Haustein and Petra Schwille. Fluorescence correlation spectroscopy: Novel variations of an established technique. *Annu. Rev. Biophys. Biomol. Struct.*, 36:151–169, 2007.

[84] Benedict Hebert, Santiago Costantino, and Paul W Wiseman. Spatiotemporal image correlation spectroscopy (STICS) theory, verification, and application to protein velocity mapping in living CHO cells. *Biophysical journal*, 88(5):3601–14, 2005. doi: 10.1529/biophysj.104.054874.

[85] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2012.

[86] Ekbert Hering and Rolf Martin, editors. *Photonik*. Springer, 2005. ISBN 9783540234388.

[87] Romey F. Heuff, Jody L. Swift, and David T. Cramb. Fluorescence correlation spectroscopy using quantum dots: advances, challenges and opportunities. *Phys. Chem. Chem. Phys.*, 9:1870–1880, 2007. doi: 10.1039/b617115j.

[88] Gerrit Heuvelman, Fabian Erdel, Malte Wachsmuth, and Karsten Rippe. Analysis of protein mobilities and interactions in living cells by multifocal fluorescence fluctuation microscopy. *European Biophysics Journal*, 38(6):813–828, 2009. doi: 10.1007/s00249-009-0499-9.

[89] Gerald C. Holst. *CCD arrays, cameras and displays*. SPIE Optical Engineering Press, 2 edition, 1998. ISBN 0-9640000-4-0.

[90] B. Hoppe, H. Meuth, M. Engels, and R. Peters. Design of digital correlation systems for low-intensity precision photon spectroscopic measurements. *lEEE Proceedings Circuits, Devices & Systems*, 148(5):267–271, 2001.

[91] B. Hoppe, H. Meuth, M. Engels, and R. Peters. Design of digital correlation systems for low-intensity precision photon spectroscopic measurements. In *IEEE Proceedings Circuits, Devices and Systems*, volume 148, pages 267–271. IET, 2001. doi: 10.1049/ip-cds:20010363.

[92] Margaret R. Horton, Felix Höfling, Joachim O. Rädler, and Thomas Franosch. Development of anomalous diffusion among crowding proteins. *Soft Matter*, 6(12):2648–2656, 2010. doi: 10.1039/B924149C.

[93] Marianne S Hromalik, Katherine S Green, Hugh T Philipp, Mark W Tate, and Sol M Gruner. The fpga pixel array detector. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 701:7–16, 2013.

[94] David A Huffman et al. A method for the construction of minimum redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[95] Jan Huisken, Jim Swoger, Filippo Del Bene, Joachim Wittbrodt, and Ernst H. K. Stelzer. Optical sectioning deep inside live embryos by selective plane illumination microscopy. *Science*, 305 (5686):1007–1009, 2004. doi: 10.1126/science.1100035.

[96] Ling Chin Hwang and Thorsten Wohland. Single wavelength excitation fluorescence cross-correlation spectroscopy with spectrally similar fluorophores: Resolution for binding studies. *The Journal of Chemical Physics*, 122(11):114708, 2005. doi: doi:10.1063/1.1862614.

[97] A. Ingargiola, R. A. Colyer, D. Kim, F. Panzeri, R. Lin, A. Gulinatti, I. Rech, M. Ghioni, S. Weiss, and X. Michalet. Parallel multispot smFRET analysis using an 8-pixel SPAD array. *Proceedings of the SPIE*, 8228:82280B–82280B–8, 2012. doi: 10.1117/12.909470.

[98] Intel Corporation. Intel 64 and IA-32 architectures software developer's manual - volume 2. `http://www.intel.ie/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-instruction-set-reference-manual-325383.pdf`. accessed 2015-01-12.

[99] Mark A. Itzler, Xudong Jiang, and Mark Entwistle. Power law temporal dependence of InGaAs/InP SPAD afterpulsing. *Journal of Modern Optics*, 59(17):1472–1480, 2012. doi: 10.1080/09500340.2012.698659.

[100] E Jakeman. Photon correlation. In *Photon correlation and light beating spectroscopy*, pages 75–149. Springer, 1974.

[101] C. Jakob, A. Th. Schwarzbacher, B. Hoppe, and R. Peters. The development of a digital multichannel correlator system for light scattering experiments. In *Irish Signals and Systems Conference, 2006. IET*, pages 99–103, 2006. ISBN 0-86341-665-9. URL http://www.electronics.dit.ie/staff/aschwarzbacher/research/issc2006multi.pdf.

[102] C Jakob, A. Th. Schwarzbacher, B. Hoppe, and R. Peters. A FPGA optimised digital Real-Time mutichannel correlator architecture. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, 2007. DSD 2007*, pages 35–42. IEEE, 2007. doi: 10.1109/DSD.2007.4341447.

[103] C. Jakob, A.T. Schwarzbacher, B. Hoppe, and R. Peters. A multichannel digital real-time correlator as single fpga implementation. In *15th International Conference on Digital Signal Processing, 2007*, pages 276–279, july 2007. doi: 10.1109/ICDSP.2007.4288572.

[104] Wesley M Johnston, JR Hanna, and Richard J Millar. Advances in dataflow programming languages. *ACM Computing Surveys (CSUR)*, 36(1):1–34, 2004.

[105] Stanislav Kalinin, Ralf KuÌünemuth, Hayk Vardanyan, and Claus A. M. Seidel. Note: A 4 ns hardware photon correlator based on a general-purpose field-programmable gate array development board implemented in a compact setup for fluorescence correlation spectroscopy. *Review of Scientific Instruments*, 83(9):096105, 2012. doi: 10.1063/1.4753994.

[106] Pakorn Kanchanawong and Clare M Waterman. Localization-based super-resolution imaging of cellular structures. *Methods in molecular biology (Clifton, N.J.)*, 1046:59–84, 2013. doi: 10.1007/978-1-62703-538-5_4.

[107] Balakrishnan Kannan, Lin Guo, Thankiah Sudhaharan, Sohail Ahmed, Ichiro Maruyama, and Thorsten Wohland. Spatially resolved total internal reflection fluorescence correlation microscopy using an electron multiplying Charge-Coupled device camera. *Analytical Chemistry*, 79(12):4463–4470, 2007. doi: 10.1021/ac0624546.

[108] P. J. Keller, A. D. Schmidt, J. Wittbrodt, and E. H.K. Stelzer. Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy. *Science*, 322(5904):1065–1069, 2008. doi: 10.1126/science.1162493.

[109] Alexander Khintchine. Korrelationstheorie der stationären stochastischen prozesse. *Mathematische Annalen*, 109(1):604–615, 1934.

[110] Donald Kinghorn. Haswell floating point performance. http://www.pugetsystems.com/blog/2013/08/26/Haswell-Floating-Point-Performance-493/, August 2012. accessed 2015-03-25.

[111] M. Kloster-Landsberg, D. Tyndall, I. Wang, R. Walker, J. Richardson, R. Henderson, and A. Delon. Note: Multi-confocal fluorescence correlation spectroscopy in living cells using a complementary metal oxide semiconductor-single photon avalanche diode array. *Review of Scientific Instruments*, 84(7):076105, 2013. doi: 10.1063/1.4816156.

[112] R H Köhler, P Schwille, W W Webb, and M R Hanson. Active protein transport through plastid tubules: velocity quantified by fluorescence correlation spectroscopy. *Journal of cell science*, 113 (Pt 22):3921–30, 2000.

[113] Z. Kojro, A. Riede, M. Schubert, and W. Grill. Systematic and statistical errors in correlation estimators obtained from various digital correlators. *Review of Scientific Instruments*, 70(12): 4487–4496, 1999. doi: 10.1063/1.1150101.

[114] David L. Kolin, Santiago Costantino, and Paul W. Wiseman. Sampling effects, noise, and photobleaching in temporal image correlation spectroscopy. *Biophysical Journal*, 90:628–639, 2006.

[115] David L Kolin, David Ronis, and Paul W Wiseman. k-space image correlation spectroscopy: a method for accurate transport measurements independent of fluorophore photophysics. *Biophysical journal*, 91(8):3061–75, 2006. doi: 10.1529/biophysj.106.082768.

[116] Leon Gordon Kraft. *A device for quantizing, grouping, and coding amplitude-modulated pulses*. PhD thesis, Massachusetts Institute of Technology, 1949.

[117] Matthias Kretz and Volker Lindenstruth. Vc: A c++ library for explicit vectorization. *Software: Practice and Experience*, 42(11):1409–1430, 2012.

[118] Oleg Krichevsky and Grégoire Bonnet. Fluorescence correlation spectroscopy: the technique and its applications. *Reports on Progress in Physics*, 65(2):251–297, 2002. doi: 10.1088/0034-4885/65/2/203.

[119] J. W. Krieger. Mapping diffusion properties in living cells. 2014.

[120] Jan Krieger. How can we get faster?, 2014.

[121] Jan W. Krieger, Jan Buchholz, Nicolas Schnellbächer, Christoph S. Garbe, and Jörg Langowski. QuickFit 3.0 – a data evaluation package for (imaging) fluorescence correlation spectroscopy. *BMC Bioinformatics*, 2015. in preparation.

[122] Jan Wolfgang Krieger and Jörg Langowski. QuickFit 3.0 – an open-source data evaluation application for biophysics. source code: `https://github.com/jkriege2/QuickFit3`, binaries: `http://www.dkfz.de/Macromol/quickfit/`, 2015.

[123] Jan Wolfgang Krieger, Anand Pratap Singh, Christoph S. Garbe, Thorsten Wohland, and Jörg Langowski. Dual-Color fluorescence Cross-Correlation spectroscopy on a single plane illumination microscope (SPIM-FCCS). *Optics Express*, 22(3):2358, 2014. doi: 10.1364/OE.22.002358.

[124] Ian Carlos Kuon. *Measuring and navigating the gap between FPGAs and ASICs*. PhD thesis, University of Toronto, 2008.

[125] Jeffrey J. Lange, Christopher J. Wood, William A. Marshall, Lucinda E. Maddera, Qingfeng E. Yu, William D. Bradford, Brian D. Slaughter, Jay R. Unruh, and Winfried Wiegraebe. Correction of bleaching artifacts in high content fluorescence correlation spectroscopy (HCS-FCS) data. *Proc. SPIE 8590, Single Molecule Spectroscopy and Superresolution Imaging VI*, 859006: 859006–859006–15, 2013. doi: 10.1117/12.2005205.

[126] Paul Langevin. Sur la théorie du mouvement brownien. *CR Acad. Sci. Paris*, 146(530-533), 1908.

[127] N. Laracuente and C. Grossman. Real-time autocorrelator for fluorescence correlation spectroscopy based on graphical-processor-unit architecture: method, implementation, and comparative studies. In *APS March Meeting Abstracts*, page H1212, March 2013.

[128] T. Laurence, S. Fore, and T. Huser. A fast, flexible algorithm for calculating correlations in fluorescence correlation spectroscopy. *Opt. Lett.*, 31(6), 2006.

[129] Hsu-Yang Lee, Hsin-Yu Lin, Jonathon D. White, and Wunshain Fann. High-speed low-cost correlator for single molecule fluorescence correlation spectroscopy. *Proceedings*, 7185(1): 71850R–71850R–8, 2009. doi: 10.1117/12.809426.

[130] Valeria Levi and Enrico Gratton. Chromatin dynamics during interphase explored by single-particle tracking. *Chromosome research : an international journal on the molecular, supramolecular and evolutionary aspects of chromosome biology*, 16(3):439–49, 2008. doi: 10.1007/s10577-008-1240-8.

[131] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. Nvidia tesla: A unified graphics and computing architecture. *Ieee Micro*, 28(2):39–55, 2008.

[132] Mingguo Liu, Chong Hu, Joe C. Campbell, Zhong Pan, and Mark M. Tashima. Reduce afterpulsing of single photon avalanche diodes using passive quenching with active reset. *IEEE Journal of Quantum Electronics*, 44(5):430–434, 2008. doi: 10.1109/JQE.2007.916688.

[133] Ping Liu, Thankiah Sudhaharan, Rosita M.L. Koh, Ling C. Hwang, Sohail Ahmed, Ichiro N. Maruyama, and Thorsten Wohland. Investigation of the dimerization of proteins from the epidermal growth factor receptor family by single wavelength fluorescence Cross-Correlation spectroscopy. *Biophysical Journal*, 93(2):684–698, 2007. doi: 10.1529/biophysj.106.102087.

[134] Ping Liu, Sohail Ahmed, and Thorsten Wohland. The f-techniques: advances in receptor protein studies. *Trends in Endocrinology & Metabolism*, 19(5):181–190, 2008. doi: 10.1016/j.tem.2008.02.004.

[135] W. Liu, J. Shen, and X. Sun. Design of multiple-tau photon correlation system implemented by fpga. In *ICESS'08. International Conference on Embedded Software and Systems, 2008*, pages 410–414. IEEE, 2008.

[136] Wei Liu, Jin Shen, and Xianming Sun. Design of Multiple-Tau photon correlation System Implemented by FPGA. In *Proceedings of The 2008 International Conference on Embedded Software and Systems (ICESS2008)*, pages 410–414. IEEE Computer Society, 2008. doi: 10.1109/ICESS.2008.7.

[137] Enterpoint Ltd. Broaddown4 user manual, issue – 2.00 draft, April 2007.

[138] Katherine Luby-Phelps, Philip E. Castle, D Lansing Taylor, and Lanni Frederick. Hindered diffusion of inert tracer particles in the cytoplasm of mouse 3t3 cells. *Proceedings of the National Academy of Sciences*, 84(14):4910–4913, 1987.

[139] D. Lumma, S. Keller, T. Vilgis, and J. O. Rädler. Dynamics of large semiflexible chains probed by fluorescence correlation spectroscopy. *Physical Review Letters*, 90(21):218301, 2003.

[140] D Magatti and F Ferri. Fast multi-tau real-time software correlator for dynamic light scattering. *Applied Optics*, 40(24):4011–21, 2001. doi: 10.1364/AO.40.004011.

[141] Davide Magatti and Fabio Ferri. 25 ns software correlator for photon and fluorescence correlation spectroscopy. *Review of Scientific Instruments*, 74(2):1135–1144, 2003. doi: 10.1063/1.1525876.

[142] D Magde, E L Elson, and W W Webb. Fluorescence correlation spectroscopy i: Conceptual basis and theory. *Biopolymers*, 13(1):1–27, 1974. doi: 10.1002/bip.1974.360130102.

[143] D Magde, E L Elson, and W W Webb. Fluorescence correlation spectroscopy. II. an experimental realization. *Biopolymers*, 13(1):29–61, 1974. doi: 10.1002/bip.1974.360130103.

[144] Douglas Magde, Watt W. Webb, and Elliot L. Elson. Fluorescence correlation spectroscopy. III. uniform translation and laminar flow. *Biopolymers*, 17(2):361–376, 1978. doi: 10.1002/bip.1978.360170208.

[145] Timo Mappes, Norbert Jahr, Andrea Csaki, Nadine Vogler, Jürgen Popp, and Wolfgang Fritzsche. The invention of immersion ultramicroscopy in 1912-the birth of nanotechnology? *Angewandte Chemie International Edition*, 51(45):11208–11212, 2012. doi: 10.1002/anie.201204688.

[146] Shelley R McRae, Christopher L Brown, and Gillian R Bushell. Rapid purification of egfp, eyfp, and ecfp with high yield and purity. *Protein expression and purification*, 41(1):121–127, 2005.

[147] Oskar Mencer. Multiscale dataflow computing. In *Multi-/Many-core Computing Systems (MuCoCoS), 2013 IEEE 6th International Workshop on*, pages 1–1. IEEE, 2013.

[148] Ralf Metzler and Joseph Klafter. The random walk's guide to anomalous diffusion: a fractional dynamics approach. *Physics Reports*, 339(1):1–77, 2000.

[149] Ralf Metzler and Joseph Klafter. The restaurant at the end of the random walk: recent developments in the description of anomalous transport by fractional dynamics. *Journal of Physics A*, 37: R161–R208, 2003.

[150] Tom K. L. Meyvis, Stefaan C. De Smedt, Patrick Van Oostveldt, and Joseph Demeester. Fluorescence recovery after photobleaching: A versatile tool for mobility and interaction measurements in pharmaceutical research. *Pharmaceutical Research*, 16(8):1153–1162, 1999. doi: 10.1023/A:1011924909138.

[151] X Michalet, RA Colyer, G Scalia, S Weiss, Oswald HW Siegmund, Anton S Tremsin, John V Vallerga, F Villa, F Guerrieri, I Rech, et al. New photon-counting detectors for single-molecule fluorescence spectroscopy and imaging. In *SPIE Defense, Security, and Sensing*, pages 803316–803316. International Society for Optics and Photonics, 2011.

[152] Gábor Mocsár, Balázs Kreith, Jan Buchholz, Jan Wolfgang Krieger, Jörg Langowski, and György Vámosi. Note: Multiplexed multiple-tau auto- and cross-correlators on a single field programmable gate array. *Review of Scientific Instruments*, 83(4):046101, 2012. doi: 10.1063/1.3700810.

[153] Norbert Mücke, Stefan Winheim, Holger Merlitz, Jan Buchholz, Jörg Langowski, and Harald Herrmann. Kinetics of intermediate filament assembly in vitro: A monte carlo simulation study. In preparation, 2015.

[154] Claus B Müller, Kerstin Weiß, Walter Richtering, Anastasia Loman, and Joerg Enderlein. Calibrating differential interference contrast microscopy with dual-focus fluorescence correlation spectroscopy. *Optics express*, 16(6):4322–4329, 2008.

[155] Jan Buchholz, Jan Wolfgang Krieger, Gábor Mocsár, Balázs Kreith, Edoardo Charbon, György Vámosi, Udo Kebschull, and Jörg Langowski. FPGA implementation of a 32x32 autocorrelator array for analysis of fast image series. *Optics Express*, 20(16):17767, 2012. doi: 10.1364/OE.20.017767.

[156] Daniel J. Needleman, Yangqing Xu, and Timothy J. Mitchison. Pin-Hole array correlation imaging: Highly parallel fluorescence correlation spectroscopy. *Biophysical Journal*, 96(12): 5050–5059, 2009. doi: 10.1016/j.bpj.2009.03.023.

[157] C. Niclass, A. Rochas, P.-A. Besse, and E. Charbon. Design and characterization of a CMOS 3-d image sensor based on single photon avalanche diodes. *IEEE Journal of Solid-State Circuits*, 40 (9):1847–1854, 2005. doi: 10.1109/JSSC.2005.848173.

[158] C. Niclass, M. Sergio, and E. Charbon. A single photon avalanche diode array fabricated in 0.35-μm cmos and based on an event-driven readout for tcspc experiments. In *Proc. SPIE*, volume 6372, page 63720S, 2006.

[159] Cristiano Niclass and Mineki Soga. A miniature actively recharged single-photon detector free of afterpulsing effects with 6ns dead time in a 0.18µm CMOS technology. In *2010 International Electron Devices Meeting*, pages 14.3.1–14.3.4. IEEE, 2010. ISBN 978-1-4424-7418-5. doi: 10.1109/IEDM.2010.5703360.

[160] Cristiano Niclass, Claudio Favi, Theo Kluter, Marek Gersbach, and Edoardo Charbon. A 128×128 Single-Photon image sensor with Column-Level 10-bit Time-to-Digital converter array . *IEEE JOURNAL OF SOLID-STATE CIRCUITS*, 43:2977–2989, 2008. doi: 10.1117/12.685974.

[161] nikie. Is there a way to detect if an image is blurry?, December 2014. URL `https://stackoverflow.com/questions/7765810/is-there-a-way-to-detect-if-an-image-is-blurry`.

[162] NVIDIA Corporation. CUDA toolkit documentation v7.0. `http://docs.nvidia.com/cuda/`. accessed 2015-03-09.

[163] NVIDIA Corporation. NVIDIA's next generation CUDA compute architecture: Kepler GK110. whitepaper. `http://www.nvidia.com/content/PDF/kepler/NVIDIA-kepler-GK110-Architecture-Whitepaper.pdf`, 2012. accessed 2015-03-09.

[164] NVIDIA Corporation. NVIDIA Pascal GPU Architecture. `http://blogs.nvidia.com/blog/2015/03/17/pascal/`, March 2015. accessed 2015-08-19.

[165] Jun Ohta. *Smart CMOS Image Sensors And Applications*. CRC Press, 2008. ISBN 9780849336812.

[166] Kunle Olukotun and Lance Hammond. The future of microprocessors. *Queue*, 3(7):26–29, 2005.

[167] M. Ormö, A. B. Cubitt, K. Kallio, L. A. Gross, R. Y. Tsien, and S. J. Remington. Crystal structure of the aequorea victoria green fluorescent protein. *Science*, 273(5280):1392–1395, 1996. doi: 10.1126/science.273.5280.1392.

[168] Timo Ottenstein. *A new objective for high resolution imaging of Bose-Einstein condensates*. PhD thesis, Diploma thesis, University of Heidelberg, 2006.

[169] Juan Mata Pavia, Martin Wolf, and Edoardo Charbon. Measurement and modeling of microlenses fabricated on single-photon avalanche diode arrays for fill factor recovery. *Optics Express*, 22(4): 4202, 2014. doi: 10.1364/OE.22.004202.

[170] Jean Perrin. L'agitation moléculaire et le mouvement brownien. *Comptes rendus hebdomadaires des séances de l'académie des sciences*, 146:967–970, 1908.

[171] Jean Perrin. Mouvement brownien et réalité moléculaire. *Annales de Chimie et de Physique*, 18: 5–104, 1909.

[172] M. Jean Perrin and F. Soddy. *Brownian movement and molecular reality*. Taylor & Francis, 1910.

[173] Zdeněk Petrášek and Petra Schwille. Precise measurement of diffusion coefficients using scanning fluorescence correlation spectroscopy. *Biophysical journal*, 94(4):1437–1448, 2008.

[174] Eric Quinnell, EE Swartzlander, and Carl Lemonds. Floating-point fused multiply-add architectures. In *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on*, pages 331–337. IEEE, 2007. URL `https://www.lib.utexas.edu/etd/d/2007/quinnelle60861/quinnelle60861.pdf`.

[175] Ivan Rech, Stefano Marangoni, Daniele Resnati, Massimo Ghioni, and Sergio Cova. Multipixel single-photon avalanche diode array for parallel photon counting applications. *Journal of Modern Optics*, 56(2-3):326–333, 2009. doi: 10.1080/09500340802318309.

[176] Jonas Ries, Salvatore Chiantia, and Petra Schwille. Accurate determination of membrane dynamics with Line-Scan FCS. *Biophysical Journal*, 96(5):1999–2008, 2009. doi: 10.1016/j.bpj.2008.12.3888.

[177] Jonas Ries, Zdeněk Petrášek, Anna J García-Sáez, and Petra Schwille. A comprehensive framework for fluorescence cross-correlation spectroscopy. *New Journal of Physics*, 12(11):113009, 2010. doi: 10.1088/1367-2630/12/11/113009.

[178] Jonas Ries, Zdeněk Petrášek, Anna J García-Sáez, and Petra Schwille. A comprehensive framework for fluorescence cross-correlation spectroscopy. *New Journal of Physics*, 12(11):113009, 2010.

[179] R. Rigler, Ü. Mets, J. Widengren, and P. Kask. Fluorescence correlation spectroscopy with high count rate and low background: analysis of translational diffusion. *European Biophysics Journal*, 22(3):169–175, 1993. doi: 10.1007/BF00185777.

[180] D Rohr. Alice tpc online tracking on gpgpu based on kalman filter. *University of Heidelberg, Diploma Thesis*, 2010.

[181] Christian Michael Roth. *Untersuchung in lebenden Zellen mit Hilfe neuer Methoden der Einzelmolekülfluoreszenzspektroskopie*. Doktorarbeit, Universität Heidelberg, 2006. URL `http://archiv.ub.uni-heidelberg.de/volltextserver/volltexte/2006/6661/pdf/Dissertation_Christian_M._Roth.pdf`.

[182] Qiaoqiao Ruan, Melanie A Cheng, Moshe Levi, Enrico Gratton, and William W Mantulin. Spatial-temporal studies of membrane dynamics: scanning fluorescence correlation spectroscopy (SFCS). *Biophysical journal*, 87(2):1260–7, 2004. doi: 10.1529/biophysj.103.036483.

[183] Bahaa E. A. Saleh and Malvin Carl Teich. *Funddamentals of Photonics*. John Wiley & Sons, 1991. ISBN 0-471-83965-5.

[184] Bahaa EA Saleh and Mavlin Carl Teich. Fundamentals of photonics. 2007.

[185] Jagadish Sankaran, Nirmalya Bag, Rachel Susan Kraut, and Thorsten Wohland. Accuracy and precision in Camera-Based fluorescence correlation spectroscopy measurements. *Analytical Chemistry*, 85(8):3948–3954, 2013. doi: 10.1021/ac303485t.

[186] K. Schätzel. New concepts in correlator design. *Inst. Phys. Conf. Ser.*, 77:175–84, 1985.

[187] Klaus Schätzel. New concepts in correlator design. *Institute of Physics Conference Series*, 77: 175–184, 1985.

[188] Klaus Schätzel. New concepts in correlator design. *Inst. Phys. Conf. Ser.*, 77:175–184, 1985.

[189] Klaus Schätzel. Noise on photon correlation data: I. autocorrelation functions. *Quantum Opt.*, 2: 287–305, 1990. doi: 10.1088/0954-8998/2/4/002.

[190] Klaus Schätzel, Martin Drewel, and Sven Stimac. Photon correlation measurements at large lag times: improving statistical accuracy. *Journal of Modern Optics*, 35(4):711–718, 1988.

[191] E. Schaub. F2Cor: fast 2-stage correlation algorithm for FCS and DLS. *Opt. Express*, 20(3): 2184–2195, 2012.

[192] Petra Schwille, Ulrich Haupts, Sudipta Maiti, and Watt W. Webb. Molecular dynamics in living cells observed by fluorescence correlation spectroscopy with one- and Two-Photon excitation. *Biophysical Journal*, 77(4):2251–2265, 1999. doi: 10.1016/S0006-3495(99)77065-7.

[193] Peter Seitz and Albert J. P. Theuwissen. *Single-Photon Imaging*. Springer, 2011. ISBN 9783642184420.

[194] Mohammad Reza Selim and Mohammed Ziaur Rahman. Carrying on the legacy of imperative languages in the future parallel computing era. *Parallel Computing*, 40(3):1–33, 2014.

[195] Teruo Shimmen and Etsuo Yokota. Cytoplasmic streaming in plants. *Current Opinion in Cell Biology*, 16(1):68–72, 2004. doi: 10.1016/j.ceb.2003.11.009.

[196] Osamu Shimomura, Frank H. Johnson, and Yo Saiga. Extraction, purification and properties of aequorin, a bioluminescent protein from the luminous Hydromedusan,Aequorea. *Journal of Cellular and Comparative Physiology*, 59(3):223–239, 1962. doi: 10.1002/jcp.1030590302.

[197] Roman Shusterman, Sergey Alon, Tatyana Gavrinyov, and Oleg Krichevsky. Monomer dynamics in double- and Single-Stranded DNA polymers. *Physical Review Letters*, 92(4):048303, 2004. doi: 10.1103/PhysRevLett.92.048303.

[198] Roman Shusterman, Tatyana Gavrinyov, and Oleg Krichevsky. Internal dynamics of superhelical DNA. *Physical review Letters*, 100:098102, 2008. doi: 10.1103/PhysRevLett.100.098102.

[199] H. Siedentopf and R. Zsigmondy. Über Sichtbarmachung und Größenbestimmung ultramikoskopischer Teilchen, mit besonderer Anwendung auf Goldrubingläser. *Annalen der Physik*, 315(1):1–39, 1902. doi: 10.1002/andp.19023150102.

[200] Anand Pratap Singh. Light sheet based fluorescence correlation and cross-correlation spectroscopy for quantitative measurements of bio-molecules in live cells, 2014. URL `http://scholarbank.nus.edu.sg/handle/10635/118300?show=full`.

[201] Anand Pratap Singh, Jan Wolfgang Krieger, Jan Buchholz, Edoardo Charbon, Jörg Langowski, and Thorsten Wohland. The performance of 2D array detectors for light sheet based fluorescence correlation spectroscopy. *Optics Express*, 21(7):8652, 2013. doi: 10.1364/OE.21.008652.

[202] Daniel R. Sisan, Richard Arevalo, Catherine Graves, Ryan McAllister, and Jeffrey S. Urbach. Spatially resolved fluorescence correlation spectroscopy using a spinning disk confocal microscope. *Biophysical Journal*, 91(11):4241–4252, 2006. doi: 10.1529/biophysj.106.084251.

[203] Robert Sjöback, Jan Nygren, and Mikael Kubista. Absorption and fluorescence properties of fluorescein. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 51(6):L7–L21, 1995.

[204] Rebecca C. Somers, Moungi G. Bawendi, and Daniel G. Nocera. CdSe nanocrystal based chem-/bio- sensors. *Chemical Society Reviews*, 36(4):579, 2007. doi: 10.1039/B517613C.

[205] Alfred Spector and David Gifford. The space shuttle primary computer system. *Communications of the ACM*, 27(9):872–900, 1984.

[206] Robert Stephens. A survey of stream processing. *Acta Informatica*, 34(7):491–541, 1997.

[207] M Suomalainen, M Y Nakano, S Keller, K Boucke, R P Stidwill, and U F Greber. Microtubule-dependent plus- and minus end-directed motilities are competing processes for nuclear targeting of adenovirus. *The Journal of cell biology*, 144(4):657–72, 1999. doi: 10.1083/jcb.144.4.657.

[208] Simon M. Sze and Kwok K. Ng. *Physics of Semiconductor Devices*. Wiley-Interscience, 3 edition, 2006. ISBN 9780471143239.

[209] Life Technologies. Alexa fluor®dyes spanning the visible and infrared spectrum-section 1.3 | life technologies, . URL http://probes.invitrogen.com/handbook/sections/0103.html. accessed 2014-11-10.

[210] Life Technologies. The alexa fluor dye series-note 1.1, . URL http://www.lifetechnologies.com/de/de/home/references/molecular-probes-the-handbook/technical-notes-and-product-highlights/the-alexa-fluor-dye-series.html. accessed 2015-05-20.

[211] Life Technologies. Fluorescence quantum yields and lifetimes for alexa fluor dyes-table 1.5, . URL http://www.lifetechnologies.com/de/de/home/references/molecular-probes-the-handbook/tables/fluorescence-quantum-yields-and-lifetimes-for-alexa-fluor-dyes.html. accessed 2015-05-20.

[212] Life Technologies. Qdot probes- quantum dots nanotechnology, . URL http://www.lifetechnologies.com/de/de/home/brands/molecular-probes/key-molecular-probes-products/qdot.html. accessed 2014-11-10.

[213] Life Technologies. Qdot®525 streptavidin conjugate, . URL http://www.lifetechnologies.com/order/catalog/product/Q10143MP. accessed 2014-11-10.

[214] Life Technologies. Tetraspeck®Microspheres, 0 1µm, fluorescent blue/green/orange/dark red, . URL https://www.lifetechnologies.com/order/catalog/product/T7279. accessed 2015-06-17.

[215] Life Technologies. Fluospheres fluorescent microspheres, 2005. URL https://tools.lifetechnologies.com/content/sfs/manuals/mp05000.pdf. accessed 2015-05-20.

[216] Life Technologies. Qdot itk carboxyl quantum dots, 2007. URL http://tools.lifetechnologies.com/content/sfs/manuals/mp19020.pdf. accessed 2015-05-20.

[217] B. Tieman, S. Narayanan, A. Sandy, and M. Sikorski. MPICorrelator: a parallel code for performing time correlations. *Nucl. Inst. Meth. A*, 649(1):240–242, 2011.

[218] Simone Tisa, Nick Bertone, and Franco Zappa. A 32×32 photon counting camera. *Optik & Photonik*, 6(2):43–46, 2011. doi: 10.1002/opph.201190329.

[219] Salvatore Tudisco, Francesco Musumeci, Luca Lanzano, Agata Scordino, Simona Privitera, Angelo Campisi, Luigi Cosentino, Giovanni Condorelli, Paolo Finocchiaro, Giorgio Fallica, Salvatore Lombardo, Massimo Mazzillo, Delfo Sanfilippo, and Emilio Sciacca. A new generation of SPAD—Single-Photon avalanche diodes. *IEEE Sensors Journal*, 8(7):1324–1329, 2008. doi: 10.1109/JSEN.2008.926962.

[220] György Vámosi, Nina Baudendistel, Claus-Wilhelm von der Lieth, Nikoletta Szalóki, Gábor Mocsár, Gabriele Müller, Péter Brázda, Waldemar Waldeck, SÃ¡ndor Damjanovich, Jörg Langowski, and Katalin Tóth. Conformation of the c-Fos/c-Jun complex in vivo: A combined FRET, FCCS, and MD-Modeling study. *Biophysical Journal*, 94(7):2859–2868, 2008. doi: 10.1529/biophysj.107.120766.

[221] Patrick D van der Puije. Appendix d: The modified huffman data compression code. *Telecommunication Circuit Design, Second Edition*, pages 401–403, 2002.

[222] Marco Vitali, Danilo Bronzi, A Krmpot, Stanko Nikolic, F Schmitt, Cornelia Junghans, Simone Tisa, Thomas Friedrich, Vladana Vukojevic, Lars Terenius, et al. A single-photon avalanche camera for fluorescence lifetime imaging microscopy and correlation spectroscopy. 2014.

[223] A. H. Voie, D. H. Burns, F. A. Spelman, A. H. Voie, D. H. Burns, and F. A. Spelman. Orthogonal-plane fluorescence optical sectioning: Three-dimensional imaging of macroscopic biological specimens. *Journal of Microscopy*, 170(3):229–236, 1993. doi: 10.1111/j.1365-2818.1993.tb03346.x.

[224] M. von Smoluchowski. Zur kinetischen theorie der brownschen molekularbewegung und der suspensionen. *Annalen der Physik*, 326(14):756–780, 1906. doi: 10.1002/andp.19063261405.

[225] Malte Wachsmuth, Waldemar Waldeck, and Jörg Langowski. Anomalous diffusion of fluorescent probes inside living cell nuclei investigated by spatially-resolved fluorescence correlation spectroscopy. *Journal of Molecular Biology*, 298(4):677–689, 2000. doi: 10.1006/jmbi.2000.3692.

[226] M. Wahl, I. Gregor, M. Patting, J. Enderlein, et al. Fast calculation of fluorescence correlation data with asynchronous time-correlated single-photon counting. *Opt. Express*, 11(26):3583–3591, 2003.

[227] Michael Wahl, Ingo Gregor, Matthias Patting, and Jörg Enderlein. Fast calculation of fluorescence correlation data with asynchronous time-correlated single-photon counting. *Optics Express*, 11 (26):3583–3591, 2003.

[228] Michael Wahl, Hans-Jürgen Rahn, Ingo Gregor, Rainer Erdmann, and Jörg Enderlein. Dead-time optimized time-correlated photon counting instrument with synchronized, independent timing channels. *Review of Scientific Instruments*, 78(3):033106, 2007. doi: 10.1063/1.2715948.

[229] Robert A Walker and Donald E Thomas. A model of design representation and synthesis. In *Proceedings of the 22nd ACM/IEEE Design Automation Conference*, pages 453–459. IEEE Press, 1985.

[230] Christopher S Wallace. A suggestion for a fast multiplier. *Electronic Computers, IEEE Transactions on*, (1):14–17, 1964.

[231] Vince Weaver and Jack Dongarra. Can hardware performance counters produce expected, deterministic results. In *3rd Workshop on Functionality of Hardware Performance Monitoring*, pages 1–11, 2010.

[232] Matthias Weiss, Hitoshi Hashimoto, and Tommy Nilsson. Anomalous protein diffusion in living cells as seen by fluorescence correlation spectroscopy. *Biophysical Journal*, 84(6):4043–4052, 2003. doi: 10.1016/S0006-3495(03)75130-3.

[233] Matthias Weiss, Markus Elsner, Fredrik Kartberg, and Tommy Nilsson. Anomalous subdiffusion is a measure for cytoplasmic crowding in living cells. *Biophysical Journal*, 87(5):3518–3524, 2004. doi: 10.1529/biophysj.104.044263.

[234] Jerker Widengren, Rudolf Rigler, and Ülo Mets. Triplet-State monitoring by fluorescence correlation spectroscopy. *Journal of Fluorescence*, 4(3):255–259, 1994.

[235] Albert X. Widmer and Peter A. Franaszek. A dc-balanced, partitioned-block, 8b/10b transmission code. *IBM Journal of research and development*, 27(5):440–451, 1983.

[236] P. Wiseman. Image correlation spectroscopy: mapping correlations in space, time, and reciprocal space. *Methods in enzymology*, 518:245–67, 2013. doi: 10.1016/B978-0-12-388422-0.00010-8.

[237] Paul W. Wiseman, Jeffrey A. Squier, and Kent R. Wilson. Dynamic image correlation spectroscopy (ICS) and two-color image cross-correlation spectroscopy (ICCS): concepts and application. In Jose-Angel Conchello, Carol J. Cogswell, Andrew G. Tescher, and Tony Wilson, editors, *Three-Dimensional and Multidimensional Microscopy: Image Acquisition Processing VII*, volume 3919 of *SPIE Proceedings*, pages 14–20. SPIE, 2000. doi: 10.1117/12.384193.

[238] Paul W. Wiseman, Claire M. Brown, Donna J. Webb, Benedict Hebert, Natalie L. Johnson, Jeff A. Squier, Mark H. Ellisman, and A. F. Horwitz. Spatial mapping of integrin interactions and dynamics during cell migration by image correlation microscopy. *Journal of Cell Science*, 117(Pt 23):5521–5534, 2004. doi: 10.1242/jcs.01416.

[239] T. Wocjan, J. Krieger, O. Krichevsky, and J. Langowski. Dynamics of a fluorophore attached to superhelical dna: Fcs experiments simulated by brownian dynamics. *Phys. Chem. Chem. Phys.*, 11(45):10671–10681, 2009. doi: 10.1039/B911857H.

[240] Tomasz Wocjan, Jan Krieger, Oleg Krichevsky, and Jörg Langowski. Dynamics of a fluorophore attached to superhelical DNA: FCS experiments simulated by brownian dynamics. *Physical Chemistry Chemical Physics*, 11(45):10671, 2009. doi: 10.1039/B911857H.

[241] Thorsten Wohland, Xianke Shi, Jagadish Sankaran, and Ernst H. K. Stelzer. Single plane illumination fluorescence correlation spectroscopy (SPIM-FCS) probes inhomogeneous three-dimensional environments. *Optics Express*, 10(8):10627–10641, 2010. doi: 10.1364/OE.18.010627.

[242] Joachim Wuttke. lmfit–ac/c++ routine for levenberg-marquardt minimization with wrapper for least-squares curve fitting. version 5.1, retrieved on 2015-04-07., 2013. URL `http://apps.jcns.fz-juelich.de/lmfit`.

[243] Xilinx, inc. Xilinx glossary. `http://www.xilinx.com/company/terms.htm`, . accessed 2015-05-29.

[244] Xilinx, inc. What is a FPGA. `http://www.xilinx.com/fpga/`, . accessed 2015-04-13.

[245] Xilinx, inc. Virtex-4 family overview. `http://www.xilinx.com/support/documentation/data_sheets/ds302.pdf`, September 2009.

[246] Xilinx, inc. Virtex-4 family overview. `http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf`, August 2010.

[247] Xilinx, inc. Virtex-ii pro and virtex-ii pro x platform fpgas: Complete data sheet. http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf, June 2011.

[248] Xilinx, inc. Virtex-6 CXT family data sheet. http://www.xilinx.com/support/documentation/data_sheets/ds153.pdf, February 2011.

[249] Xilinx, inc. Virtex-6 fpga memory interface solutions. http://www.xilinx.com/support/documentation/ip_documentation/ds186.pdf, March 2011.

[250] *Virtex-6 FPGA DSP48E1 Slice*. Xilinx, inc., February 2011.

[251] Xilinx, inc. XST user guide for virtex-6, spartan-6, and 7 series devices. http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_1/xst_v6s6.pdf, March 2011.

[252] Xilinx, inc. Virtex-6 family overview. http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf, January 2012.

[253] Xilinx, inc. Virtex-6 FPGA configurable logic block - user guide. http://www.xilinx.com/support/documentation/user_guides/ug364.pdf, February 2012.

[254] Xilinx, inc. Ml605 hardware user guide. http://www.xilinx.com/support/documentation/boards_and_kits/ug534.pdf, October 2012.

[255] Xilinx, inc. Virtex-6 FPGA memory resources - user guide. http://www.xilinx.com/support/documentation/user_guides/ug363.pdf, February 2014.

[256] Y. Yang, J. Shen, W. Liu, and Y. Cheng. Digital Real-Time correlator implemented by field programmable gate array. In *CISP'08. Congress on Image and Signal Processing, 2008*, volume 1, pages 149–151. IEEE, 2008.

[257] Guohong Zhang, Vanessa Gurtu, and Steven R Kain. An enhanced green fluorescent protein allows sensitive detection of gene transfer in mammalian cells. *Biochemical and biophysical research communications*, 227(3):707–711, 1996.

[258] Ming Zhao, Lei Jin, Bo Chen, Yao Ding, Hui Ma, and Dieyan Chen. Afterpulsing and its correction in fluorescence correlation spectroscopy experiments. *Applied Optics*, 42(19):4031, 2003. doi: 10.1364/AO.42.004031.

[259] Xiang Zhu and Dianwen Zhang. Efficient parallel levenberg-marquardt model fitting towards real-time automated parametric imaging microscopy. *PloS one*, 8(10):e76665, 2013.

[260] Richard A. Zsigmondy. Properties of colloids, 1925. URL http://www.nobelprize.org/nobel_prizes/chemistry/laureates/1925/zsigmondy-lecture.pdf. accessed 2014-03-08.

Acknowledgements

Many people directly or indirectly contributed to my PhD project and this thesis, whom I would like to thank.

- Foremost I would like to thank Prof. Jörg Langowski and Prof. Udo Kebschull for providing the opportunity to work on this interesting project. I would also like to thank them for their support, fruitful discussions, and supervision of my thesis.
- I also want to express my gratitude to the Helmholtz International Graduate School for Cancer Research and the German Cancer Research Center (DKFZ) for their financial support during this thesis.
- I would like to thank our collaboration partner Prof. Edoardo Charbon (EPFL / TU Delft) for providing the hardware basis of this work, *i.e.*, the SPAD arrays, invaluable support, and many helpful discussions.
- I would like to thank Prof. Ana J. García-Sáez for being a member of my TAC-committee.
- A very special thanks goes to my colleague and friend Jan Wolfgang Krieger for building the microscope, teaching QuickFit3 new tricks on a regular basis, numerous discussions and intensive proof-reading. Thank you for a great time, extraordinary trips, and 'cool' BBQs.
- I would like to thank Norbert Mücke for lively discussions concerning my project, his projects, our projects, proofreading of my thesis, and the challenge for 2016! ;-)
- I want to thank Gabriele Müller for providing all kinds of cells.
- A huge thank you also goes to all the other people that I collaborated with during this thesis: Samuel Burri, Heiko Engel, Frederik Grüll, Balázs Kreith, Yuki Maruyama, Gábor Mocsár, and György Vámosi.
- For the nice atmosphere and for being valuable companions during and beyond the daily routine, I would like to thank Stefan Winheim, Felix Rettig, Alexander Gansen, Kathrin Tegeler, Tabea Elbel, Vera Böhm, and Thomas Kühn.
- I would like to thank all the people who proofread parts of my thesis and helped me with invaluable advice during the writing.
- I want to thank all the members of our group who made my time in B040 worthwhile.
- A very special thank you goes to all my patient family and friends I've unfortunately seen far too rarely in recent times. Thank you for showing me that there is still another life outside of the office.
- I deeply thank my parents Christine and Fried (†2007) and my sister Britta. Without their unconditional support and appreciation, I would not be who I am today.
- Finally, I want to thank Sabrina Pleier for all her patience, love, and constant support during the last years. Thank you for believing in me and cheering me up every single day.