FPGA-Based Phase Stabilization of an Optical Lattice

Bachelor Thesis

Author(s):
Schnuck, Maria Sophie

Publication date:
2023

Permanent link:
https://doi.org/10.3929/ethz-b-000643872

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

FPGA-Based Phase Stabilization of an Optical Lattice

Bachelor's Thesis

Maria Sophie Schnuck

mschnuck@student.ethz.ch

Quantum Optics Group
ETH Zürich

Supervisors:
Marius Gächter (Quantum Optics Group)
Samuel Jele (Quantum Optics Group)
Dr. Abdulkadir Akin (Quantum Device Lab)
Prof. Lukas Novotny (Photonics Laboratory)

October 14, 2023

Abstract

This bachelor thesis presents the implementation of a real-time phase stabilization system for an optical lattice, utilizing Fast Fourier Transform (FFT) and acousto-optic steering. The chosen hardware platform is a Field-Programmable Gate Array (FPGA), offering low-latency and flexibility in implementation.
A prototype of the stabilization system was developed on a PC to gain insights into the system's characteristics. While this prototype demonstrates the feasibility of the stabilization approach, it highlights the limitation with a feedback rate of 20 Hz in effectively controlling higher-frequency lattice phase fluctuations. The final FPGA design successfully calculates the lattice phase within 6.22 μs. Operating at a feedback rate of 1 kHz, this implementation achieves stabilization to within $\pm 10^{-2}\pi$ of the setpoint. Furthermore, this work provides a detailed analysis of the phase measurement error based on simulations of the implemented calculation method. The measurements are expected to be accurate up to $\pm 10^{-2}\pi$ with potential for significant improvement when using a camera with higher intensity resolution.
These results are relevant for future experiments with ultracold atoms in optical lattices because phase stabilization enables precise control and alignment of lattices relative to each other, which can be used to access and perform measurements on single atoms of the lattice. The next steps towards implementation of the phase stabilization in the real quantum experiment include improving the accuracy of the phase measurement and extending the current approach to two-dimensional lattices.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As a platform for analog quantum simulators, ultracold atoms trapped in optical lattices have gained increasing attention in research. The principle of analog quantum simulators is to approximate a real system as well as possible by a well-controlled quantum system in order to solve complex many-body problems. An ultracold quantum gas interacts with a light crystal, effectively mimicking how electrons behave in solid-state materials. The lattice acts as a potential on the ultracold atoms, attracting them to either maxima or minima of light intensity.[1]
One advantage of optical lattices is their many degrees of freedom, as different lattice configurations can be realized by varying intensity and lattice spacing or utilizing structures of multiple superimposed lattices (superlattices).[2]

The Lattice group in the Quantum Optics group at ETH Zurich has realized such an experimental setup with optical lattices. For future experiments, it is of great interest to gain access to individual atoms in the lattice. As proposed in reference [3], two superimposed accordion lattices can be used as a method to increase the distance between atoms. Accordion lattices are lattices with variable spacing. In this case, the lattice spacing has to be controllable over a wide range of values because the lattice constant must be repeatedly doubled and halved during the expansion process. Such a range of lattice constants can be realized by changing the angle between the interfering beams that form the lattice. However, in such a setup, it cannot be avoided for the beams to travel long distances before they interfere.[3]
On the path to the optical lattice the phases of the laser beams are subject to drifts caused, for example, by small fluctuations in temperature, pressure or mechanical noise. As the relative phase between the beams changes, the position of the maxima in the lattice shifts, which results in an unstable lattice. Since the experiment requires precise alignment of two lattices relative to each other, stabilizing the relative phase between the beams is necessary.
This can be achieved with acousto-optic steering of the laser beams.[3] To measure the phase of the optical lattice, images of the lattice are taken with a camera and analyzed with the Discrete Fourier Transform (DFT).[4]

The objective of this work is to implement the proposed stabilization method for a one-dimensional lattice. This involves extracting the lattice phase in real-time using the Fourier transform and generating a control signal for acousto-optic steering. The target is to stabilize lattice drifts of frequencies up to 100 Hz.
In a first step, the algorithm is tested on a PC-based prototype to gain further insight into the behavior of the system and to provide a proof of principle for the stabilization mechanism. In a second step, the design is implemented on a Field-Programmable Gate Array (FPGA) to achieve the maximum feedback rate, equal to the image acquisition rate of the camera.
Chapter 2 covers the theoretical background necessary to understand the system. Following this, Chapter 3 explains the methods used during implementation and testing. The implementation on the PC and the FPGA will be described in Chapter 4, followed by the test results in Chapter 5.

Chapter 2

Background

2.1 One-Dimensional Optical Lattices

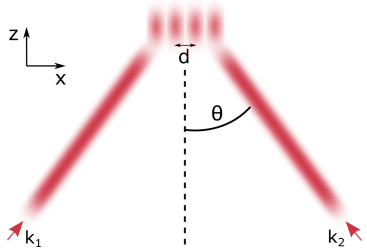

Figure 2.1: Example of a one-dimensional lattice in the xz-plane formed by two laser beams

A one-dimensional lattice can be created using two laser beams of the same wavelength that meet at an angle, as shown in Figure 2.1. For the purpose of this work, it is sufficient to consider the beams as plane waves in the xz-plane with intensity I of the following form

$$E_i(x, z, t) = \sqrt{I} \cos\left(\pm k_x x + k_z z - \omega t + \phi_i\right) \tag{2.1}$$

where $i = 1, 2$ for the rays travelling in positive and negative x directions, respectively. k_x and k_z are the components of the wave vector in the x and z directions, ω is the angular frequency, and ϕ_i is the phase of the plane wave.
The total field is then given by the sum of the two plane waves, resulting in an interference pattern of the form

$$I(x) = 2I \cos^2\left(k_x x + \frac{\Delta\phi}{2}\right) \tag{2.2}$$

$\Delta\phi = \phi_1 - \phi_2$ being the relative phase between the beams, also called lattice phase in the following. The lattice has a spacing of

$$d = \frac{\lambda}{2 \sin\theta} \tag{2.3}$$

with λ the wavelength of the laser and θ the angle under which the beams interfere.[4]

2.2 Lattice Phase Measurement

As can be seen in Equation 2.2 the lattice phase $\Delta\phi$ influences the position of the maxima within the lattice. Drifts of this lattice phase with a frequency of up to several Hertz can often not be avoided. This leads to undesired instability of the lattice, which makes active stabilization necessary. To generate the feedback for this stabilization, the lattice phase can be extracted using the Fourier transform of the lattice.
For the one-dimensional Fourier transform, the following convention is used

$$\mathcal{F}[f(x)](q) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x) e^{-iqx}\, dx \tag{2.4}$$

The transform of the intensity distribution in Equation 2.2 is given by

$$\mathcal{F}[I(x)](q) = I\sqrt{\frac{\pi}{2}} \left(e^{i\Delta\phi}\delta(q - 2k_x) + e^{-i\Delta\phi}\delta(q + 2k_x) + 2\delta(q) \right) \tag{2.5}$$

The lattice phase can now be extracted by calculating the phase of the Fourier transform at $q = 2k_x$:

$$\Delta\phi = \arctan\left(\frac{\Im\left(\mathcal{F}[I(x)](2k_x)\right)}{\Re\left(\mathcal{F}[I(x)](2k_x)\right)} \right) \tag{2.6}$$

For this purpose, the value of k_x need not be known in advance. The value of k_x can be extracted by determining the maximum absolute value in the positive half of the transform, provided that the component at $q = 0$ is filtered out.[4]

2.3 Fast Fourier Transform

The Fast Fourier Transform (FFT) is an optimized algorithm to compute the DFT of a signal with length N, assuming $N = 2^n$. The general formula for the DFT of a signal $x_0, ..., x_N - 1$ is

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi ikn/N} \tag{2.7}$$

for $k = 0, ..., N - 1$.
The FFT algorithm recursively splits the signal into its even and odd components using the identity

$$\begin{aligned} X_k &= \sum_{n=0}^{N/2-1} x_{2n} e^{-4\pi ikn/N} + e^{-2\pi ik/N} \sum_{n=0}^{N/2-1} x_{2n+1} e^{-4\pi ikn/N} \\ X_{k+N/2} &= \sum_{n=0}^{N/2-1} x_{2n} e^{-4\pi ikn/N} - e^{-2\pi ik/N} \sum_{n=0}^{N/2-1} x_{2n+1} e^{-4\pi ikn/N} \end{aligned} \tag{2.8}$$

for $k = 0, ..., N/2 - 1$.
This reduces the computation time from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$.

2.4 Spectral Leakage and Windowing

For simplicity, in the following the Fourier transform of a function of time will be considered, the principles are the same for functions of space. As described in the previous section the DFT only samples the signal at N points, and thus, only over a finite interval ΔT. Because the DFT assumes a periodic function, the function is implicitly extended periodically.
This circumstance does not pose an issue when the transform time ΔT aligns as an integer multiple of the fundamental period T of the signal. In that case, the values at the edges of the transform window are the same and, as a consequence, there is no discontinuity at the signal edges.
In contrast to that, if ΔT is not a multiple of T, the periodic extension causes jumps at the edges of the sampling window (see Figure 2.2).

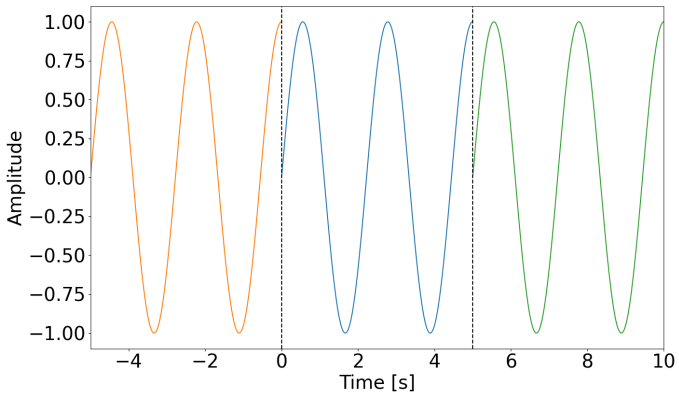

Figure 2.2: Periodic extension with boundary discontinuities; ($\Delta T = 5$ s is not an integer multiple of the period $T = \frac{20}{9}$ s); original signal (blue), periodic extension (orange and green) causes discontinuities at $t = 0$s and $t = 5$s

As a consequence, high frequency components are added to the signal, typically much higher than the Nyquist rate. They are aliased into lower frequencies, resulting in a smeared-out version of the original spectrum, a phenomenon called spectral leakage. Spectral leakage causes the ideal δ-peaks of the sine signal in Fourier space to spread into wider peaks. This effect is shown in Figure 2.3.[5]

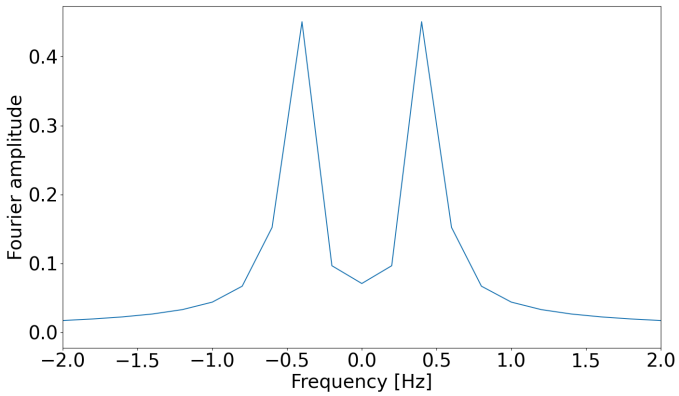

Figure 2.3: FFT of a signal with frequency leakage; the δ-peaks of the sine-functions widen due to discontinuities at the edges of the sampling window

Spectral leakage can be reduced by multiplying the sampled signal with a window function. Windowing reduces the discontinuities as the window function approaches zero at the boundaries. A large number of window functions for different use cases exists, the most popular for sinusoidal signals being the Hanning and the Hamming window. For this application, the Hanning window proved to reduce the error due to spectral leakage the most among several windows tested.
The Hanning window has the form

$$w[n] = 0.5 - 0.5\cos\left(\frac{2\pi n}{N}\right) \tag{2.9}$$

for $n = 0, ..., N-1$.
This window is suitable for this application because of its good side lobe suppression (meaning little ripple close to the main lobe) and high frequency resolution (closely spaced Fourier components can be distinguished).[5]

2.5 PID Control

PID control is a control technique widely used in industry. It aims to correct the error $e(t)$ between the setpoint and the measured signal by generating an output signal $u(t)$ that depends on a P (proportional) part, an I (integral) part and a D (derivative) part. In case of a discrete signal, the output is calculated as follows

$$u[n] = K_p e[n] + K_i \sum_{j=0}^{n} e[n] + K_d(e[n] - e[n-1]) \tag{2.10}$$

where K_p, K_i and K_d are parameters that must be tuned to the particular system.
The proportional part allows the output to react to errors quickly, but cannot eliminate the steady-state error because the proportional correction approaches zero as the error decreases. To correct this steady-state error, the integral part must be added, which ensures that the error goes to zero as the system reaches steady-state. The derivative part causes the output to respond more strongly to a rapidly changing error signal, and is therefore sensitive to noise in the signal. For this reason, the derivative part is set to zero in this work, meaning that only a PI controller is used.

Chapter 3

Methods

3.1 Test Setup

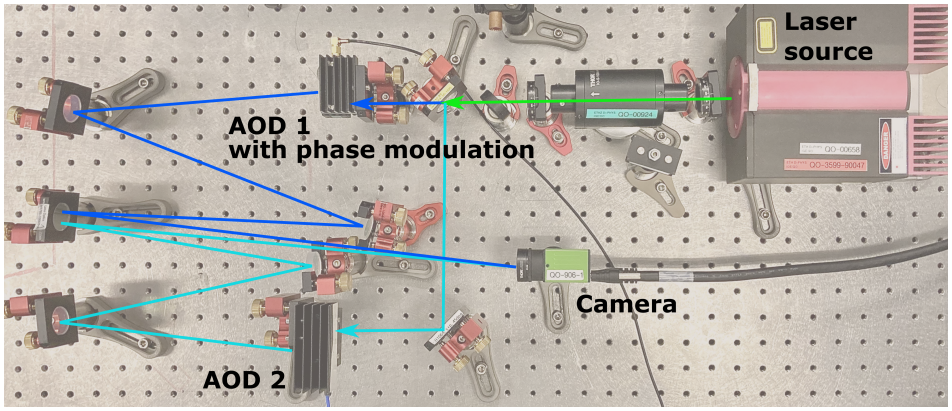

Figure 3.1: Laser setup with beam paths; common beam path (green), beam 1: phase modulated (dark blue), beam 2: not modulated (light blue)

To test the stabilization mechanism, a lattice is generated and captured with a camera as shown in Figure 3.1. The laser used has a wavelength of 1064 nm. With a beam splitter two beams are created, each of which then passes through an Acousto-Optic Deflector (AOD).
The AODs receive a sinusoidal analog input signal whose frequency determines the deflection angle (which in this case is held constant). The phase of the input signal determines the phase of the outgoing beam, so the relative phase can be changed by phase modulation at one of the AOD's inputs. In order to generate the analog signals a function generator with an external port for modulation is used.
The first order deflections of the AODs get reflected by a series of mirrors before they hit the camera. The resulting lattice constant d is measured to be 28.1 µm. This matches the expected spacing of the image on the camera in the new experiment of the Lattice group. By applying Equation 2.3, the angle at which the beams intersect was determined to be approximately 1.08°.
An exemplary image and the calculated FFT is presented in Figures 3.2 and 3.3. The Fourier peaks are positioned at $\pm 0.0356 \frac{2\pi}{\mu m}$, as one would expect with lattice spacing $d = \frac{1}{0.0356} \mu m = 28.1 \mu m$.

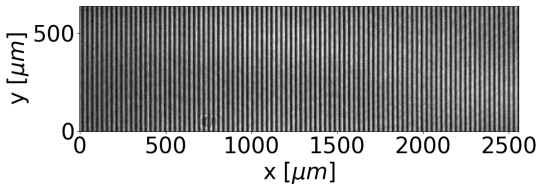

Figure 3.2: Lattice image with 128x512 pixels equivalent to 640x2560 μm; bright lines correspond to high intensity and dark lines to low intensity of light

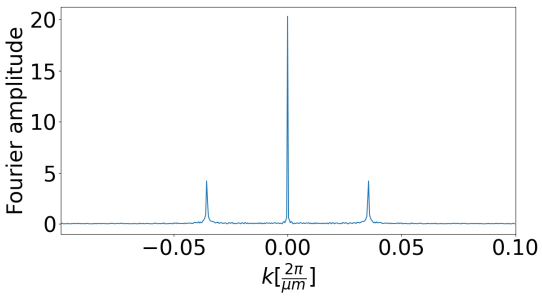

Figure 3.3: Fourier spectrum of a row of the lattice image

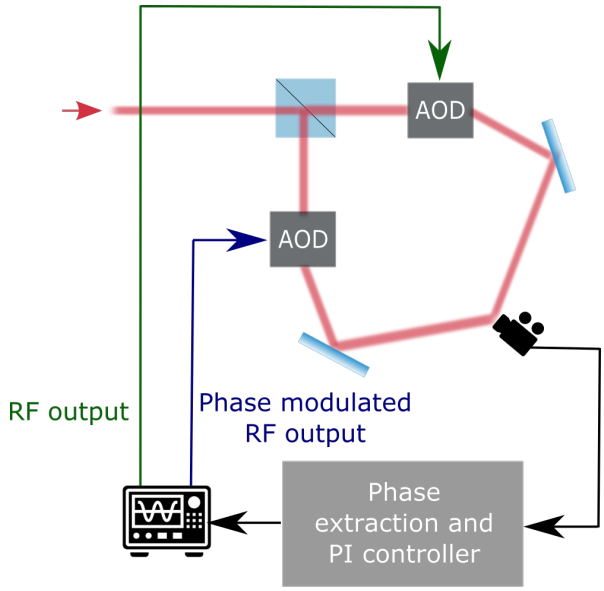

Figure 3.4: Test setup; overview of signal flow: a phase signal for the function generator is calculated from the camera images, by modulating the input of one AOD, the relative phase of the beams can be controlled

The goal of this project is to generate a signal for the function generator that determines the phase modulation. A schematic drawing of the high-level setup is shown in Figure 3.4.

In an initial version, the feedback signal is generated using a PC. This serves as a prototype to gain more insights into the behavior of the system and potential problems. For this PC-based design, the camera signals are processed using the frame grabber MicroEnable 5 marathon ACL by Basler and the corresponding driver software. The calculation of the FFT and lattice phase are implemented in Python. The results are sent to the PI controller, whose output is converted to an analog signal using the System Demonstration Platform (SDP) board and the EVAL-AD5791SDZ Digital-to-Analog Converter (DAC) board described in Section 3.2.4. At the same time, the results are displayed in the Graphical User Interface (GUI).

The final version is then implemented on an FPGA, which allows a higher feedback rate and serves as an embedded solution for the experiment. The FPGA used is described in Section 3.2.2. Image processing and control signal generation are directly integrated into the Programmable Logic (PL) of the FPGA. The digital feedback is sent to the AD5791 DAC that outputs an analog signal.
In addition, with the PmodDA3 DAC described in Section 3.2.3 an analog output signal corresponding to the lattice phase is generated. In this manner, the phase stabilization can be monitored with an oscilloscope when using the FPGA. As part of a previous student project, a GUI was implemented to communicate to the FPGA via Ethernet.[6] This allows the camera settings and PI parameters to be configured and sent to the FPGA.

3.2 Hardware

3.2.1 Camera

Figure 3.5: GO-5000M-PM camera by JAI; front and back view with two CL ports [7]

The camera used to capture the lattice is JAI's GO-5000M-PMCL. It is a monochrome area scan camera with a CMOS sensor offering 2560 x 2048 pixel resolution and 5 µm pixel dimension. Area scan cameras can capture two-dimensional images, in contrast to line scan cameras that acquire single lines of pixels and typically have much higher frame rates. The camera has two Camera Link (CL) ports and is powered via Power over Mini Camera Link because this powering method offers the highest frame rates. Additionally, the camera supports selecting a region of interest and binning options.[7]
The image is sent in 3-tap mode, meaning three pixels along the x-axis (having the same y-coordinate) are sent in parallel. The camera has an intensity resolution of 8 bits. The camera receiver that converts the raw camera signal to the intensities of the three pixels on the FPGA was implemented in a previous student project and is reused for this work.[8]
The pixel clock rate can be set to 84.99 MHz, 72.85 MHz, or 48.57 MHz. As this defines how fast pixels can be sent, the highest possible clock rate, 84.99 MHz, is chosen. The maximum frame rate depends on the region of interest, which is defined by the width and height of the transmitted image. A detailed formula can be found in reference [7]. Table 4.1 summarizes some important values.

Height (pixels)	Width (pixels)	Maximum frame rate (frames per second(fps))
2048	2560	47.8
	1024	71.8
	512	71.8
1024	2560	95.1
	1024	142.7
	512	142.7
512	2560	187.7
	1024	281.6
	512	281.6
256	2560	365.7
	1024	548.8
	512	548.8
128	2560	695.4
	1024	1043.8
	512	1043.8
64	2560	1267.4
	1024	1901.1
	512	1901.1

Table 3.1: Camera speed for different image dimensions; using a pixel clock of 84.99 MHz and 3-tap acquisition mode

Since the goal is to stabilize phase drifts of up to 100 Hz, a frame rate of at least 1000 fps is required. To achieve this, the image height is set to 128 pixels resulting in a maximum frame rate of 1043.8 fps. The width does not have such a strong effect on the frame rate and is set to 512 pixels, as this roughly corresponds to the width of the beam on the image (see Section 5.1.1 for more detail on the effect of different image widths).
For this work, which focuses on one-dimensional lattices, the image height could even be decreased further, since only one line of the image is needed. However, because the implementation will eventually need to be extended to two-dimensional lattices, the image height is not reduced further than 128 pixels.
The camera supports sending a test image that consists of a horizontal ramp in intensity. This feature is used for testing and debugging.

In summary, every time the camera is powered up it has to be configured by sending the following commands over CL:

- **CLCF = 2**, the pixel clock is set to high (84.99 MHz)
- **WTC = 512**, the image width is set to 512 pixels
- **HTL = 128**, the image height is set to 128 pixels
- **OFC = 1256**, the offset in x-direction is set, for example to 1256 pixels
- **OFL = 904**, the offset in y-direction is set, for example to 904 pixels
- **AR = 962**, the frame period is set to 962 µs, corresponding to a frame rate of 1040 fps
- **TPN = 0 or 1**, the image mode is set to normal mode or to test image mode

3.2.2 FPGA

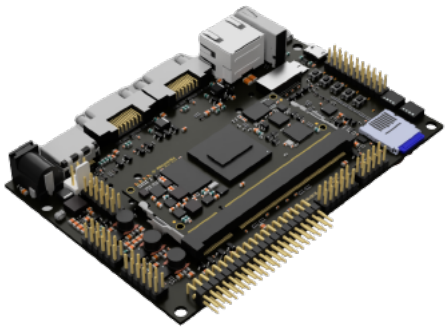

Figure 3.6: FPGA platform: Mars EB1 with Mars XU3 module by Enclustra [9][10]

In order to correct phase drifts of up to 100 Hz, the goal of this project is a feedback loop running at a rate of at least 1 kHz. On an FPGA the image processing steps can run in parallel, whereas on a computer they need to be executed sequentially. By using an FPGA, the output latency can thus be reduced. This additionally enables more complex algorithms that could be useful for two-dimensional lattices. For instance, it would be possible to transform multiple lines of an image while the image is being transmitted.
In addition, the reconfigurable PL allows easy customization of the image processing algorithm for further development. This makes FPGA a suitable hardware platform for the project.

The specific device used is the Mars XU3 FPGA module in combination with the Mars EB1 base board by Enclustra (see Figure 3.6).
The Mars XU3 is a System-on-Chip (SoC) module embedding the Xilinx Zynq UltraScale+ MPSoC. It combines PL and Processing System (PS) (including a 64-bit quad-core Arm Cortex-A53).[9] The PL consists of:

- Look-Up Tables (LUTs): combinatorial logic is implemented using predefined truth tables
- DSP48 slices: prebuilt 25-bit and 18-bit multiplier circuitry
- Block Random Access Memory (BRAM): user-defined RAM for storing and passing values

The Mars EB1 base board provides two Mini Camera Link ports (supporting Power over Mini Camera Link), an Ethernet RJ45 connector, Micro USB 2.0, and I/O-pins that are used for the DACs.[10]

3.2.3 PmodDA3

Figure 3.7: PmodDA3: 16-bit resolution DAC board [11]

Digital Interface – J1	Signal
1	CS
2	DIN
3	LDAC
4	SCLK
5	GND
6	DVDD
Digital Interface – J3	
1	GND
2	AVDD

Figure 3.8: PmodDA3 interface connector signal descriptions [11]

Digilent's PmodDA3 is a 16-bit resolution DAC board with an output range of 0 V to 2.5 V. The operating voltage is set to 3.3 V. A slightly modified Serial Peripheral Interface (SPI) protocol is used for communication, using the pins listed in Figure 3.8 (only interface J1 is used).
The *Chip Select (CS)* pin has to be driven low while the 16 data bits are sent on the *DIN* pin. When the *LDAC* pin is logic low, the output signal is updated. In this application, the output should be updated with each input value sent. For this reason, the *LDAC* pin is connected to ground. The SPI clock is transmitted via the *SCLK* pin during the transaction. *GND* and *DVDD* pins are connected to ground and operating voltage respectively.[11]

3.2.4 EVAL-AD5791SDZ

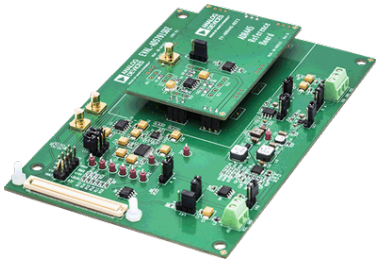

Figure 3.9: EVAL-AD5791SDZ: 20-bit resolution DAC board [12]

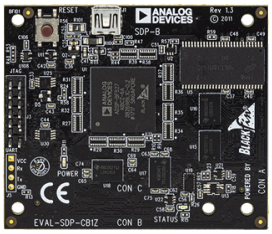

Figure 3.10: SDP-B Evaluation Board: used for communication between the PC and EVAL-AD5791SDZ [13]

The EVAL-AD5791SDZ evaluation board serves as a DAC board to generate an analog feedback signal. Its main component is Analog Devices' 20-bit resolution AD5791 DAC. Using the on-board amplifier, the reference voltage is buffered and inverted resulting in an output voltage range of −10 V to 10 V. The DAC supports a sampling rate of up to 1 MSPS.[12]

Communication to the DAC is based on SPI protocol. As with the PmodDA3, the LDAC pin is connected to ground. The pins SDO (communication from slave to master), Reset and CLR (clear) are not used.
Before the DAC register (the register containing the output value) is written to, it has to be configured by writing to the control register. For this application, the value 0b0010 0000 0000 0000 0001 0010 is written, corresponding to a reference input voltage of up to 10 V and binary coding for the DAC register values instead of 2's complement representation (see Figure 3.11).

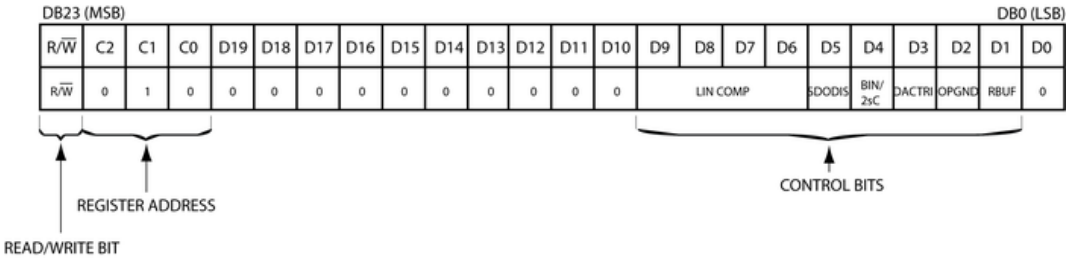

Figure 3.11: Control register of the AD5791 DAC chip [12]

When sending values from the PC to the DAC, an additional board is used to generate the SPI signals. The SDP-B evaluation board (see Figure 3.10) has a USB communication interface and can be connected directly to the EVAL-AD5791SDZ. To communicate with the SDP board, Analog Devices' Analysis, Control, Evaluation Software is installed on the PC. It can be controlled via Python. However, it only allows an update rate of 20 Hz. Higher update rates lead to a delay in the reaction of the output signal.

3.2.5 Function Generator

To generate the analog input signals for the AODs, the Siglent SDG 2122X Function Generator is used. Both outputs are set to sine waves with a frequency of 120 MHz (determining the defelection angle) and an amplitude of 370 mVpp. The phase modulation of one output is activated. It is controlled by the analog input channel of the function generator, which has a range of ±6 V and 0° to 360°. A voltage of −6 V measured at the input of the function generator corresponds to output phase of 0°, whereas a voltage of 6 V corresponds to 360°. Examining the lattice phase behavior when changing the function generator input from 6 V to −6 V, a jump in lattice phase was observed. Therefore, no rewinding at the input voltage boundaries is possible with the current function generator.

3.3 PI Tuning

An online tuner application was used to tune the parameters of the controller.[14] In order to characterize the system, a step was generated at the output of the DAC and the resulting lattice phase measured. This data was passed to the tuner, which fits a first-order model to the data and calculates the appropriate PI parameters. The parameters can be scaled to adjust the responsiveness of the controller. However, the obtained values were only used as a starting point since some fine tuning was necessary to achieve the desired stability.

For the PC-based design, the parameters $K_p = 0.5$ and $K_i = 0.1$ are used. The proportional part is relatively small compared to the integral part because, on the one hand, the lattice phase can have a large steady-state error. On the other hand, the feedback rate in the PC-based design is of the same order as the phase fluctuations, which means that the proportional value must be set conservatively to avoid oscillations and instability.

The FPGA-based design does not suffer from this limitation. The output value is updated at a rate of 1 kHz, which allows a slightly increased proportional parameter relative to the integral parameter. Using the method described above, the values $K_p = 0.43$ and $K_i = 0.016$ were chosen, as they result in a fast and stable response (see Section 5.3.2 for results).

Chapter 4

Implementation

The following chapter emphasizes the key components of the implementation. The initial version is based on a PC, but as it primarily serves as a prototype, it is not extensively optimized for low latency. Greater emphasis is placed on the second phase, namely the implementation on an FPGA.

4.1 PC-Based Prototype

The stabilization of the lattice on the PC is based on a Python application. Figure 4.1 outlines the structure of the system. On the top level, the application is implemented with PyQt, a tool for building cross-platform desktop applications. Several subprocesses are managed by the main application:

- **Frame grabber management**: control the acquisition settings of the frame grabber and manage memory resources for continuous image acquisition
- **Image callback**: executed for each received image: includes reading frame grabber memory, calculating FFT, finding the maximum value and calculating the phase of the Fourier component
- **PI control**: calculate the output voltage from the phase
- **Digital-to-analog conversion**: set configuration of DAC chip and send the calculated voltage to the DAC
- **Graphical User Interface (GUI)**: manage user inputs and update display

Each subprocess will be described in the following sections.

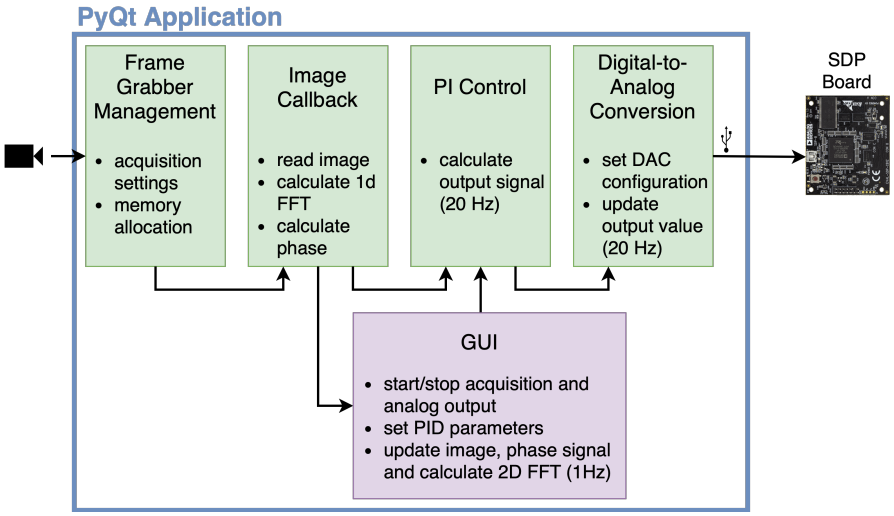

Figure 4.1: High-level block diagram of the PC-based design

4.1.1 Frame Grabber Management

This process consists of two phases: initialization and continuous acquisition.
In the initialization phase, first of all, a connection to the frame grabber is established. Following that, the acquisition settings are communicated to the frame grabber. This mainly includes the image dimension, camera sensor type, as well as tap mode (3-tap in this case). These settings are read from an initialization file that has to be updated if one of the settings is changed. Apart from that, memory resources on the frame grabber are blocked. They are used to store the images during continuous acquisition.
As soon as the image acquisition starts, two steps are repeated: At the location the image pointer currently points to, the newest image is stored along with some metadata. After that, the image pointer is moved to the next memory block to prepare for the next image.
Basic C++-based functions to communicate to the frame grabber and manage the memory pointer are provided by Basler as part of the Framegrabber Software Development Kit (SDK).

4.1.2 Image Callback

For each received image, the image callback function is executed. It comprises three steps: fetching the image, calculating the FFT, and from that the lattice phase.
First of all, the memory buffer associated with the received image number is accessed and the stored data is converted into an image array. Next, the center line of the image is multiplied element-wise with the Hanning window, of which the FFT is then calculated. Next, the result of the FFT is filtered: only the Fourier components with indices ≥ 30 (to filter out the peak at $q = 0$) and ≤ 256 (to filter out the negative half of the transform) are considered for the next step. Finally, the phase of the maximum Fourier component is returned.
To calculate this phase *numpy.arctan2()* function is used, which has the advantage that the phase is also defined if the real part of the Fourier component is zero. Additionally, in contrast to *numpy.arctan()* (that has an output within $\pm\frac{\pi}{2}$), its output range is $\pm\pi$, which matches the actual range of the lattice phase.
Still, this can lead to discontinuities when the lattice phase is close to π or $-\pi$. To adjust for this, the calculated value is compared to the previous phase value. If the absolute value of the difference is larger than 4, the phase value is adjusted by adding or substracting 2π. Subsequently, the output range is effectively extended beyond $\pm\pi$.
This callback function can be executed at up to 1 kHz. Above this rate, certain images are not processed.

4.1.3 PI Control

The PI output is calculated according to Equation 2.10, with $K_d = 0$. Next, the output is clamped to a range of $\pm 6V$.
As the analog signal can only be updated at a rate of 20 Hz while the phase calculation runs at 1 kHz, the PI controller uses the average of all phases calculated since the last output update for calculation.
One additional factor to be considered is the sign of the PI output. Depending on which AOD is phase modulated the effect of the output on the lattice phase changes sign. To account for this, the PI control output needs to be inverted. Alternatively, the phase modulation can be applied to the other output of the function generator.

4.1.4 Digital-to-Analog Conversion

For communication between the EVAL-AD5791SDZ and the PC, Analog Devices' 'Analysis, Control, Evaluation' (ACE) Software is used. It communicates to the SDP-B board via serial-USB, that then generates the SPI signals for the EVAL-AD5791SDZ. This however, proved to be the bottleneck of the system in terms of feedback rate. The ACE software only allows updating the analog output at 20 Hz.
The operation using the SDP board is straightforward: As described in Section 3.2.4, in order to use the DAC, first of all, the control register has to be initialized. After that, the values received from the PI module can be directly passed on to the DAC register.

4.1.5 Graphical User Interface

One of the main goals of the PC-based prototype is to gain more insight into the system. For real-time monitoring a GUI has been implemented, displaying the most important information (see Figure 4.2). This includes the current camera image, the time evolution of the lattice phase and the two-dimensional FFT of the image. In addition, it provides a control interface to start and stop image acquisition and analog output, set the PID parameters and store the phase signal.
While for the lattice phase calculation a one-dimensional FFT is implemented, the two-dimensional FFT allows additional insights into the system. The main purpose is to detect whether the camera axis is aligned with the lattice. This is important as a misaligned lattice results in a lower Fourier peak and a less accurate phase measurement. Figure 4.3 highlights the difference between an aligned and an unaligned camera in the two-dimensional transform. The tilt of the lattice relative to the image boundaries causes a rotation in Fourier space.
To avoid delay, the GUI is only updated at a rate of 1 Hz. In particular the calculation of the two-dimensional FFT cannot be included in the image callback function as this causes too much delay. Thus, the two-dimensional FFT is only calculated when the GUI is updated (at 1 Hz).

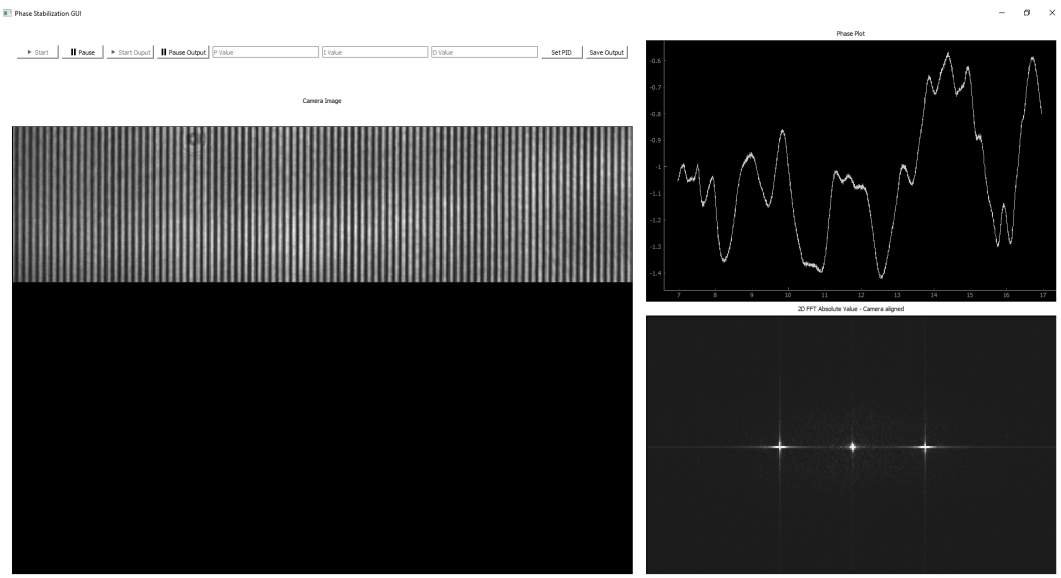

Figure 4.2: GUI for the PC-based design; upper left: control bar to start and stop image acquisition, analog output, set PID parameters and save phase signal to file; below that: camera image; upper right: lattice phase over time; lower right: two-dimensional FFT of the camera image

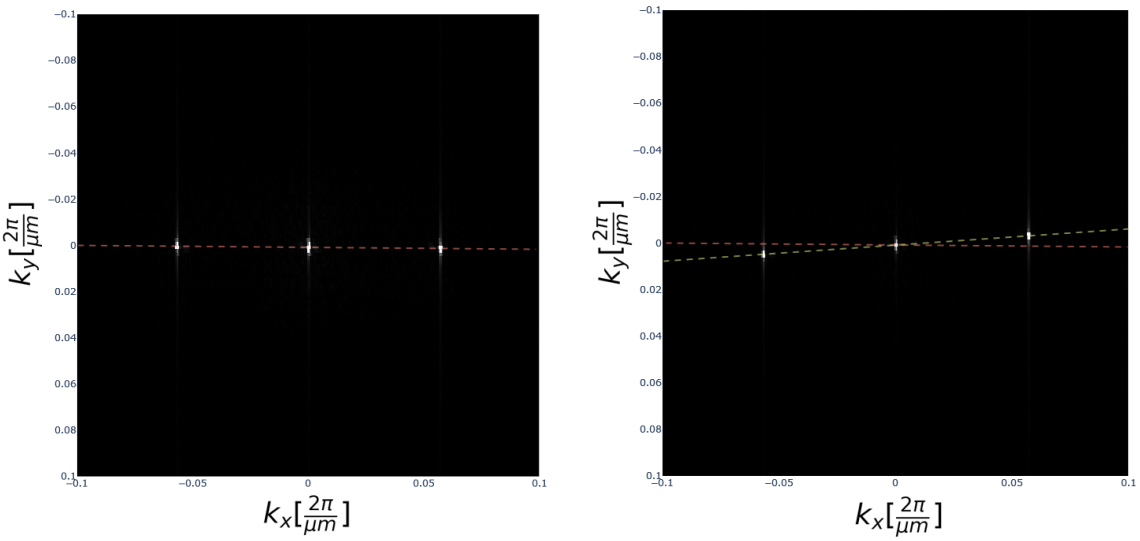

Figure 4.3: Comparison of 2D-FFT for a lattice aligned with the camera and a tilted lattice; left: Fourier transform of a lattice aligned with the camera axis; right: Fourier transform of a lattice tilted by 4.2° relative to the image axis; the maxima corresponding to the lattice spacing are visibly rotated off the x-axis for the tilted lattice

4.2 FPGA-Based Design

4.2.1 Overview

In the course of several previous projects, a camera interface and an interface to a host PC were developed.[6][8][15] The interfaces are used in this design with only small adjustments. Connections for a second camera are available but not in use in the current implementation. The high-level block design highlighting the overall architectue is presented in Figure 4.4.

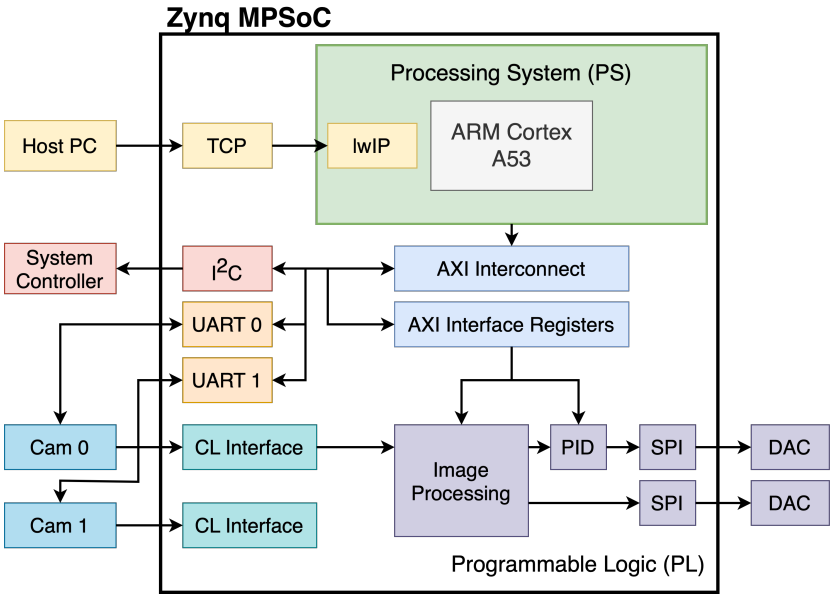

Figure 4.4: High-level block design of the FPGA-based design; interfaces to the host PC, the on-board system controller are managed with an AXI Interconnect; the camera interface is implemented on PL

The main focus of this project is the image processing block, as well as the PID controller and SPI interface. As depicted in Figure 4.5 the signal passes through several stages in the image processing block:

- **First In, First Out (FIFO) Input Wrapper**: multiply the pixel values with the Hanning window function and arrange the values for the FIFO
- **FIFO**: serialize the values and change clock domain
- **FFT Input Wrapper**: wrap the values of the FIFO into the format expected by the FFT module, generate valid and configuration signals
- **FFT**: calculate the FFT of the signal
- **Find Max**: find the maximum value of the FFT and pass it on to the Cordic module
- **Cordic**: calculate the phase of the Fourier component

Each of the stages is explained in more detail in the following sections.

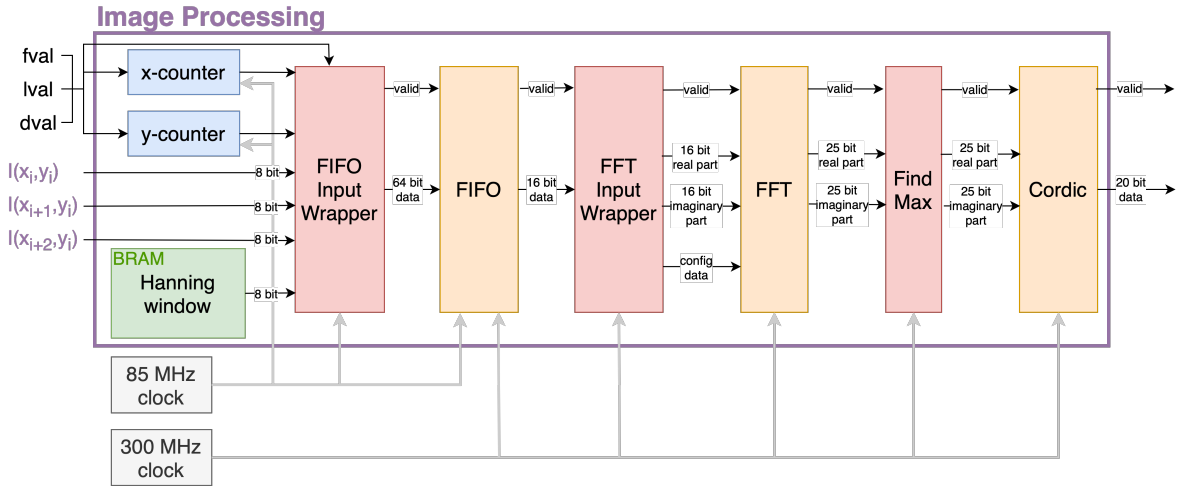

Figure 4.5: Signal flow of the image processing module; custom modules (red) and Xilinx IP cores (orange)

4.2.2 FIFO Input Wrapper

The input from the camera interface consists of the 85 MHz pixel clock, *frame valid (fval)*, *line valid (lval)*, *data valid (dval)* signals and three neighbouring pixel intensities. With two counters, signals for the x and y coordinates of the current pixels are generated. A vector containing the values of the Hanning window function with a width of 8 bits is initialized from an external file.
If the y-counter is at 64 (which corresponds to the center line of the image), the three input pixel values are multiplied with the corresponding entry of the Hanning vector. Because both factors of the multiplication have a width of 8 bits, the result is 16 bits wide. The three multiplication results are concatenated, with 16 bits of additional zeros in the end because the FIFO expects a data width of a power of 2. A *valid* signal is generated to indicate when the FIFO should read incoming values to its internal buffer.

4.2.3 FIFO

In order to serialize the pixel data a FIFO stage is used. FIFOs read the input with the read-clock's frequency and output the data with the write-clock's frequency in a first-in-first-out manner. It is possible to have differing read- and write-widths, in that case, one read-value is split into multiple write-values.
The incoming signal contains three data points per clock cycle, while the FFT module expects one data point per clock cycle as input. The FIFO can only split the data by powers of 2. Consequently, a write width of 64 bits is chosen and the signal is split into four 16 bit values that are sent sequentially at the output.
The resulting signal contains one data point per clock cycle, of which three consecutive values contain actual camera signal. Because in the FIFO Input Wrapper module 16 bits of zeros were appended, every fourth data point contains no real camera signal.
To decrease the latency of this step, a higher clock frequency is employed to output the data points. The camera interface already utilizes a 300 MHz clock generated by the PS, and by utilizing this existing clock, there is no need to generate an additional one. All subsequent modules of the image processing block are clocked with 300 MHz.

4.2.4 FFT Input Wrapper

The purpose of the FFT Input Wrapper module is to translate the received values into the correct format and a range of ± 1. Additionally, a *valid* signal, *last* signal (which is only high on the last input value) and *config* signal (determining the internal settings of the FFT module) are generated.
As explained in the previous sections, the data generated by the FIFO still contains invalid data every fourth cycle. However, the valid signal generated by the FIFO does not account for that. For this purpose, two data counters are introduced. The first counter counts all incoming data points and sets the *valid* signal to zero every fourth cycle. The second counter only counts the data points containing real values. With this second data counter, the *valid* signal is set to zero after 512 samples and the *last* signal is asserted on the last data point. The counters are reset after every frame.
The input of the FFT module is expected to be in 2's complement fixed point format with 1 integer bit and 15 fractional bits. As a consequence, the 16 bit input value is automatically interpreted to be in a range of ± 1. However, the camera signal is unsigned, meaning the values first have to be transformed to 2's complement. This can be achieved by flipping the Most Significant Bit (MSB) as explained in Figure 4.6.
As a last step, 16 bits of zeros are appended to the 16 bit real part data to represent the imaginary component of the camera signal.
Xilinx's FFT module allows changing FFT parameters during runtime by using the *config* input. This feature is not needed for the current design, thus, the *config valid* signal is set to constant zero to keep the initial configuration.

Unsigned Binary	Decimal
000	0
001	0.25
010	0.5
011	0.75
100	1
101	1.25
110	1.5
111	1.75

MSB Flip

2's Complement Binary	Decimal
100	-1
101	-0.75
110	-0.5
111	-0.25
000	0
001	0.25
010	0.5
011	0.75

Figure 4.6: Conversion of unsigned input to 2's complement by MSB flip; exemplary on 3 bit data in fixed point format with two fractional bits

4.2.5 FFT

Xilinx provides an Intellectual Property core (IP core) to calculate the FFT of an input signal. For maximum throughput, AXI (Advanced eXtensible Interface) Stream Interface is chosen. An important setting is the scaling and rounding method of the FFT. This determines if and how the intermediate results get rounded or truncated. To achieve maximum precision the unscaled option is chosen, meaning that the full bit growth is carried to the output. This results in an output width of 25 bits each for real and imaginary part. The resulting sequence $x_0, ..., x_{511}$ is provided one value per clock cycle via AXI Stream Interface. The indices 0 to 255 correspond to positive k values and the indices 256 to 511 to negative k values.
In addition, the IP core provides two options to order the output: natural and bit-reversed. Bit-reversed means that the order of the individual bits of the Fourier index is reversed. For an example of a 4-point FFT, this means the outputs are in the order: x_0, x_2, x_1, x_3 (00,10,01,11) instead of x_0, x_1, x_2, x_3 (00,01,10,11).
Bit-reversed order reduces the latency, as it is the internal order of calculation of the FFT algorithm, which is why this option is chosen.[16]

4.2.6 Find Max

To find the Fourier component corresponding to the lattice spacing, the maximum peak (apart from the peak at $q = 0$) has to be detected. For that purpose, the absolute values of the incoming FFT results need to be compared. To avoid a division, the square of the absolute value is compared instead. This is implemented with several registered stages as presented in Figure 4.7. Two additional registers are introduced to store the current maximum absolute value and the corresponding real and imaginary part.
The index counter counts the number of inputs from the FFT, bit-reverses this number to get the real index, and generates a flag that signals whether this index is in the valid range. The valid range includes all indices ≥ 30 (to filter out the peak at $q = 0$) and ≤ 255 (to filter out the negative half of the transform).
As soon as all FFT results have been received, the real and imaginary part of the detected maximum get passed on to the next stage.

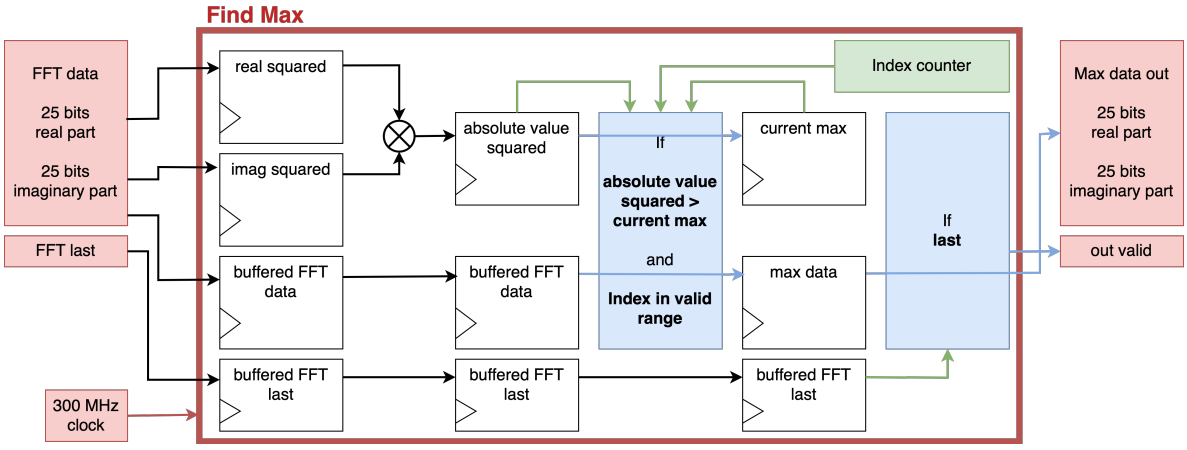

Figure 4.7: Find Max module signal flow; blue signal paths are only connected if the corresponding condition is true

4.2.7 Cordic

The Cordic IP core takes cartesian coordinates (x, y) and calculates the angle between the ray connecting this point to the origin and the positive x-axis. This is equivalent to calculating $\arctan 2(x, y)$. The difference between $\arctan 2(x, y)$ and $\arctan(x, y)$ is that $\arctan 2(x, y)$ is also defined when $x = 0$. Moreover, the output range of $\arctan 2(x, y)$ is $\pm\pi$ as explained for the PC-based design (Section 4.1.2). Consequently, the lattice phase is extracted by calculating $\arctan 2(\max_{\text{real}}, \max_{\text{imag}})$.
The output width is set to 20 bits to match the resolution of the DAC.

4.2.8 PI Control

Following the image processing stages, a module to calculate a PI control signal is necessary. As shown in Figure 4.4 the results of the image processing, meaning the calculated phases, get passed on directly to the PID module. Additionally, the PI parameters and the setpoint are sent via Ethernet from the host PC and get sent to the PID module through the PS. As mentioned in Section 2.5, the D parameter of the controller is set to zero.
The PID module is clocked with 300 MHz, however, the lattice phase is updated at a rate of 1 kHz. This is equivalent to a very long dead time in the system and would lead to saturation of the integral term. For this reason, a clock divider is added to the PID module. The clock divider generates a clock of approximately 10 kHz. The output is only updated with this clock frequency.
The PI output is calculated in three steps: Firstly, the error is calculated from the calculated lattice phase and the setpoint. Secondly, a preliminary value for the PI signal is calculated using Equation 2.10. In a last step, $0x80000$ is added to the output to adjust for the fact that $0x80000$ corresponds to an analog voltage of 0 V. Additionally, the output signal is clamped to $\pm 6V$. Simultaneously, the old integrator value and the previous error are stored for the next cycle.

Particular attention is required during the initial step, which involves computing the error signal. Because the phase is in a range of $\pm\pi$, discontinuities might occur at the boundaries of this range. It is impractical to extend the phase range as for the PC-based design (discussed in Section 4.1.3) because this would demand more bits to represent the phase.
Instead, the error calculation is adjusted slightly: In a first step, a temporary error is calculated by subtracting the input phase from the setpoint. This temporary error needs an additional bit because it is in a range of ± 8 (radian), rather than ± 4 (radian). If the absolute value of this error is larger than 5, 2π is either subtracted or added to the temporary error based on its sign, and the result is written to the final error register. The final error register maintains the original bit count, with a range of ± 4.

4.2.9 Digital-to-Analog Conversion

As described in Sections 3.2.3 and 3.2.4, two DACs are in use. They require two different modules for communication.

Analog Phase Signal

On the one hand, the PmodDA3 board is used to output an analog lattice phase signal. Digikey has created a module with an adapted SPI implementation for the PmodDA3. Only 16 bits of data can be sent through this interface, consequently, only the upper 16 bits of the calculated phase are passed on to the DAC. An additional complication is caused by the fact that the data is clocked at 300 MHz while the DAC is clocked with 50 MHz. This is resolved by adding a FIFO between the PID module and the SPI module to allow clock domain crossing.

Analog Feedback Signal

On the other hand, to generate the analog feedback signal for the function generator the EVAL-AD5791SDZ board is used. Standard SPI protocol is used for communication, however, as described in Section 3.2.4, before sending values to the DAC register, the DAC has to be initialized by a write to the control register. For this purpose, a finite state machine is added to the communication module as depicted in Figure 4.8.
The state machine starts in the *INIT* state. Every write operation consists of four steps: *PREP*, *SPI BOOT*, *SPI SEND*, and *UPDATE*. These steps set the correct value to be written, pull down the *slave select* signal, wait for a predefined number of cycles before actually sending the data, and then setting *slave select* back to high and updating internal flags.
The *PREP* state differs depending on the previous states of the state machine. In a first iteration, initial zeros are written to the DAC register (*PREP: DAC INIT VAL*). Secondly, the configuration value is written to the control register (*PREP: DAC CONFIG*). After that, the state machine repeatedly sends the current module input to the DAC register (*PREP: DAC NEXT SIGNAL*).
To improve signal quality, the clock signal is only generated if the new value is different from the previous value written to the DAC register.
Similar to the signal path for the PmodDA3, the input to the SPI module passes through a FIFO to change from the 300 MHz clock to 50 MHz used by the SPI module.

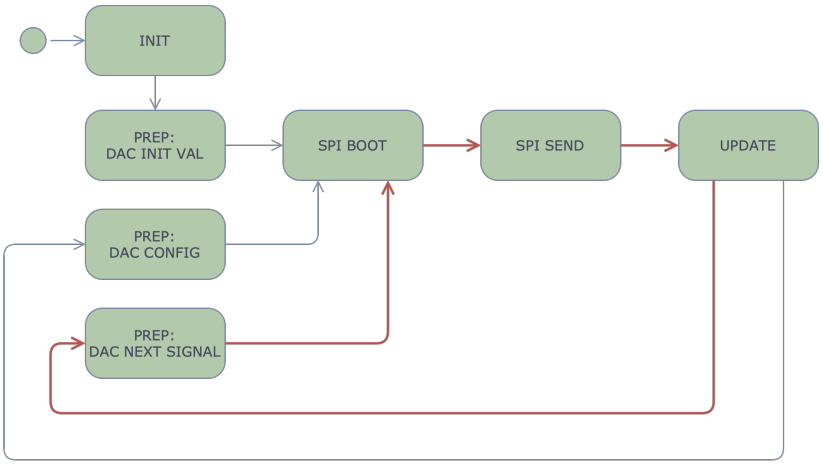

Figure 4.8: SPI finite state machine; the red loop shows the state trajectory repeated for every new DAC value

4.2.10 Utilization and Timing

A detailed breakdown of the utilization of each module on the FPGA can be found in Appendix B. In summary, the design utilizes only a minor portion of the available resources, making it feasible to duplicate the image processing block, such as adding similar processing capabilities for a second camera.
Furthermore, Table 4.1 highlights the latency of the image processing implementation. The result of the phase is available 6.217 μs after the last value of the line has been received. For comparison, transmitting a single line of the image consumes 2.012 μs (excluding the gap between two lines). As a result, multiple lines of the image can in principle be utilized for phase calculations without requiring additional resource allocation. This possibility holds potential utility for an implementation for two-dimensional lattices.
Intuitively, to analyze two-dimensional lattices, using a two-dimensional FFT is the most straight-forward approach. A naive implementation of such an approach yields a latency on the order of 10^{-2}s though, which makes it impractical considering the sampling rate should be at least 1 kHz. It is, thus, preferable to use an algorithm utilizing only one-dimensional transforms if possible.

Module	Latency (cycles)	Latency (μs)
FIFO Input Wrapper	2	0.0067
FIFO	684	2.278
FFT Input Wrapper	1	0.0033
FFT	1151	3.837
Find Max	3	0.01
Cordic	24	0.08
Total	1865	6.217

Table 4.1: Latency of image processing modules in cycles and absolute time (for a 300 MHz clock)

Chapter 5

Results

5.1 Analysis of Phase Measurement Error

Deviations between actual and calculated lattice phase can result from several sources of error that will be analyzed in the following.
First, an error is introduced by the algorithm itself, because it calculates the Fourier transform of a discrete, finite signal (instead of a continuous, periodic signal). The transform length (or image width) has a significant impact on this error. Second, the camera's intensity measurements have only finite resolution and therefore cannot reflect the intensity pattern of the lattice with perfect precision. Moreover, the measured camera signal may contain noise that distorts the result of the FFT. Finally, when working with the FPGA, an error is introduced because of the fixed-point arithmetic and finite bit size of the signals.

5.1.1 Algorithmic Error

The algorithmic error is the error inherent with the phase calculation method. It is analyzed by generating two intensity patterns of the form $I_{1/2}(x) = \cos(\frac{2\pi}{10}x + \phi_{1/2})^2$, where 10 is twice the lattice spacing in pixels, x ranges from 1 to the transform length N, and $\phi_{1/2}$ are two phase values. The algorithm is applied to calculate the phase difference between the two signals for several random phases. The calculated phase difference ϕ_{cal} is compared to the actual phase difference $\phi_{\text{real}} = \phi_1 - \phi_2$ and the result $\Delta\phi = |\phi_{\text{cal}} - \phi_{\text{real}}|$ is analyzed.
This method simulates the actual test environment because for stabilization only the relative phase between two images is of importance and not the absolute value of the calculated phase.

The results are depicted in Figure 5.1. With smaller transform sizes, the algorithmic error tends to increase due to the reduced resolution in reciprocal space (the frequency resolution Δf inversely scales with N).
Additionally, it is noteworthy that transform lengths that align as multiples of the lattice spacing exhibit significantly reduced error. This is caused by the implicit periodic extension inherent in the DFT algorithm. If the transform length is a multiple of the lattice spacing, there are almost no discontinuities at the signal edges (see Section 2.4).
However, the lattice spacing will change dynamically in the experiment, hence, the error has to be minimized over a range of lattice spacings, so this identity cannot be leveraged. For this reason, in the following sections the maximum phase deviation will be compared.

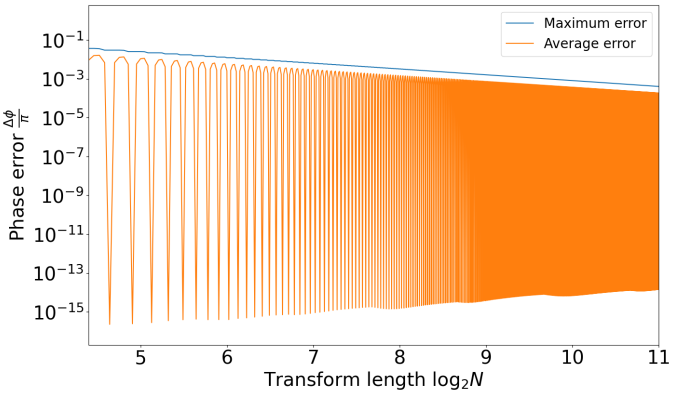

Figure 5.1: Algorithmic error without windowing; the absolute difference between calculated and actual phase values in units of π for different transform lengths; average error (orange) and maximum error (blue)

The maximum error can be reduced significantly by multiplying the signal by a window function (see Figure 5.2). Windowing reduces boundary discontinuities and spectral leakage (see Section 2.4). The Hanning function is used as it produces the smallest error for this application. All following analysis is based on the algorithm with incorporation of the Hanning window.

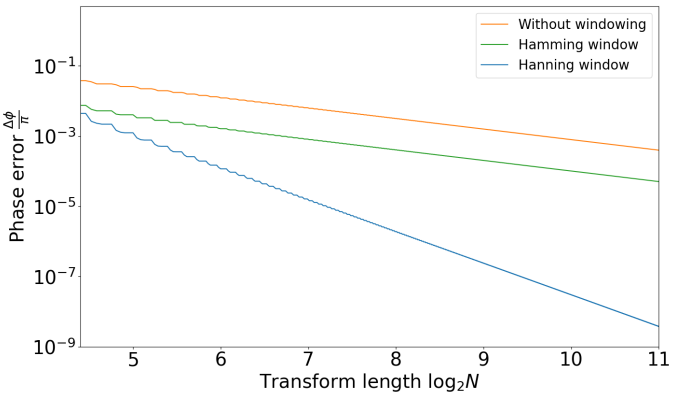

Figure 5.2: Algorithmic error with windowing; maximum phase error without windowing (orange) compared to Hamming window (green) and Hanning window (blue)

Additionally, it is of interest whether the phase error decreases or increases if the difference between the two phases is small. In reality, the phase difference between two consecutive pictures is very small, so this could influence the effective accuracy of the algorithm. However, Figure 5.3 underlines that there is no direct correlation between phase error $\Delta\phi$ and magnitude of the phase difference ϕ_{real}.

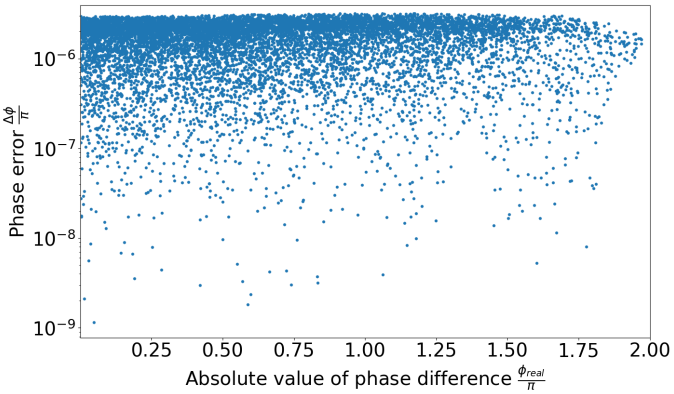

Figure 5.3: Correlation between between phase error $\Delta\phi$ and magnitude of the phase difference ϕ_{real}; transform length of $N = 2^9 = 512$; with Hanning window

5.1.2 Finite Intensity Resolution Error

As shown in Figure 5.2, the algorithmic error approaches zero with increasing transform length N. For large transform lengths, however, the finite intensity resolution error dominates. The current camera has a resolution of 8 bits or $2^8 - 1 = 255$ of which only approximately 6 bits or $2^6 = 64$ is effectively used for the lattice. By using a camera with higher intensity resolution, the error could be decreased. Figure 5.4 exemplarily compares resolutions of 6, 8 and 12 bits.

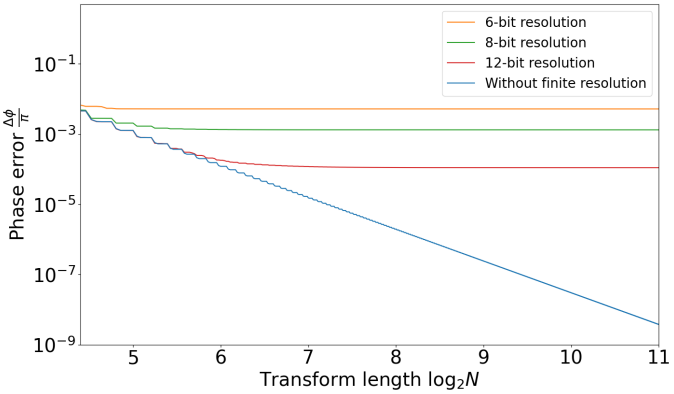

Figure 5.4: Phase error for 6-bit (orange), 8-bit (green) and 12-bit (red) intensity resolution with Hanning window, as comparison the phase error without finite intensity resolution (blue)

Not only does the image signal possess finite resolution, but the Hanning window function can also only be implemented with limited bit precision. Employing 8 bits for the Hanning function, the resulting error is negligible, on the order of $10^{-5}\pi$.

5.1.3 Image Noise

The investigation of image noise involved the introduction of a random noise signal with an amplitude of 2 and a constant offset of 10 to the intensity signal with finite bit resolution. These values were selected to align with the characteristics of the measured camera signal. The intensity signal (not considering the noise added) has an amplitude of $2^6 - 11$ because not the whole range of 8-bit values is used in the current setup. The resulting intensity pattern is $I_{1/2}(x) = (2^6 - 11) * \cos(\frac{2\pi}{10}x + \phi_{1/2})^2 + 10 + \text{random}_{[-2,2]}$.
Figure 5.5 illustrates that this addition of noise introduces an extra source of error to the phase calculation. In Figure 5.6, the same noise was applied, but this time to an intensity signal with an amplitude of $2^{12} - 11$. In effect, this reduces the relative noise amplitude. A comparison between Figures 5.5 and 5.6 clearly demonstrates that the error attributed to noise significantly diminishes with an improved signal-to-noise ratio.

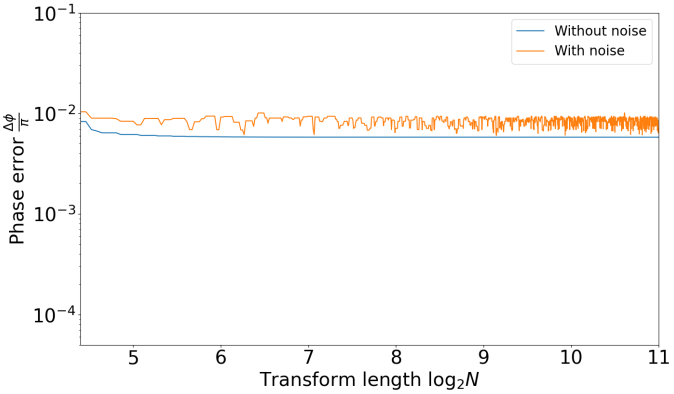

Figure 5.5: Phase error with added noise (orange) and without added noise (blue) for 6-bit intensity resolution and a noise amplitude of 2 (3.7% of the signal range) with Hanning window; this corresponds to the situation in the current setup

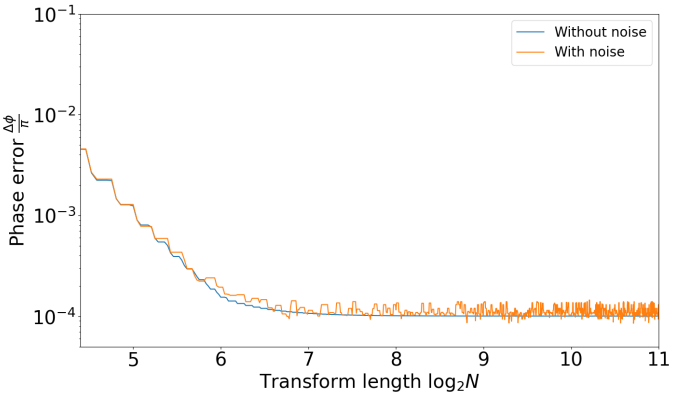

Figure 5.6: Phase error with added noise (orange) and without added noise (blue) for 12-bit intensity resolution and a noise amplitude of 2 (0.05% of the signal range) with Hanning window

5.1.4 FPGA Error

In contrast to the PC, where floating point numbers are used, the FFT on the FPGA utilizes fixed point arithmetic and finite word length. This can introduce an error, especially for very small and very large results. To quantify this error, the camera's test image mode is employed.

The test image comprises a horizontal ramp (refer to Figure 5.7), and its Fourier spectrum is depicted in Figure 5.8.

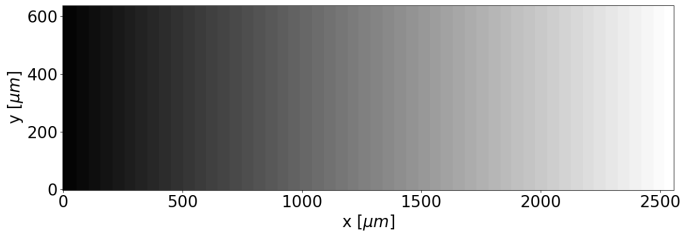

Figure 5.7: Camera test image: horizontal ramp

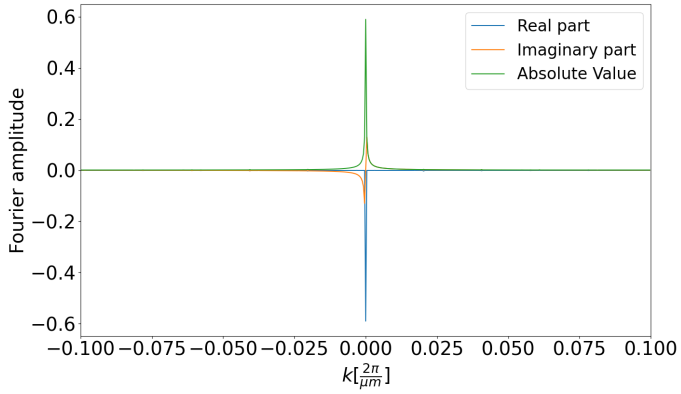

Figure 5.8: FFT of the test image

The FFT of a single line from this image is computed using both the PC and the FPGA (with values extracted using an ILA debug core). Figure 5.9 displays the absolute difference between the results obtained from the PC and the FPGA-based calculations. Notably, the imaginary component exhibits a more substantial deviation than what could be attributed solely to finite bit resolution. One possible explanation is the finite precision of the internally stored phase factors required for the transform.

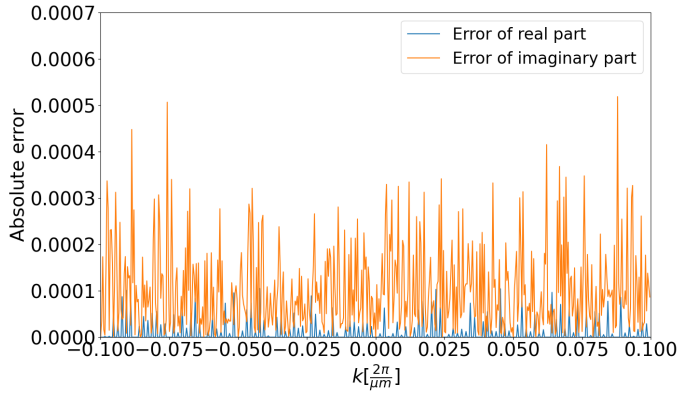

Figure 5.9: FFT deviation between PC and FPGA; absolute value of the difference between the FFT results calculated on PC and FPGA split into real and imaginary part

In principle, only the error of the phase at the maximum of the FFT is of importance and not the absolute accuracy of the FFT. The amplitude of the error is distributed evenly over the spectrum, meaning that larger Fourier components have smaller relative errors. For the phase calculation, the imaginary and real part have to be divided. Consequently, the phase error is smaller for points with large Fourier amplitudes. Since the phase is only calculated at the peak corresponding to the lattice spacing, only the phase error of large Fourier components is of interest.
In Figure 5.10 only the phase error for Fourier components larger than $\frac{1}{100}$ of the maximum amplitude are shown. This cutoff value is chosen arbitrarily but demonstrates the correlation. The phase error is below $1.2 * 10^{-3}\pi$ for this range, but for Fourier components larger than $\frac{1}{10}$ of the maximum amplitude this reduces to $10^{-4}\pi$.

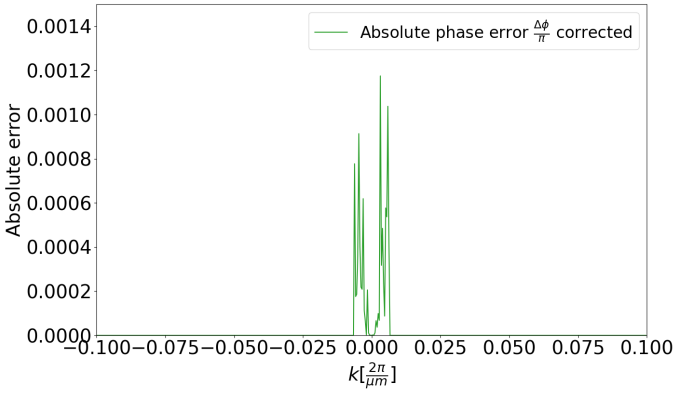

Figure 5.10: FFT phase error of Fourier components with large amplitude; the phase errors of all components smaller than $\frac{1}{100}$ of the maximum amplitude are set to zero

However, the FFT of the test image has a very different shape than that of the real lattice images. Consequently, the behavior of the FPGA-based phase error needs to be further investigated with real images to substantiate these claims. At the moment, it is not possible to transfer the images acquired with the FPGA via Ethernet. For this reason, an in-depth analysis of the phase accuracy for real lattice images is not yet possible efficiently. For this purpose, as well as general debugging and monitoring applications, it would be beneficial to implement a mechanism to transfer the images via Ethernet in future projects.

5.1.5 Summary of Error Effects

Based on the findings from the preceding sections, we can estimate the overall error under various conditions. In this discussion, we will primarily consider the algorithmic and finite-intensity errors since the error analysis for the FPGA is not yet fully robust. Additionally, the error cause by camera noise, which is of relatively low amplitude and difficult to influence, will be disregarded.

As discussed in Section 5.1.1, incorporating a Hanning window function leads to a significant reduction in error. Besides the window function, two key factors that can be adjusted are the intensity resolution of the camera and the image width (or equivalently, the transform length). Figure 5.11 illustrates the phase error for various combinations of these parameters.

The results indicate that a minimum intensity resolution of 10 bits and a transform length of $2^6 = 64$ result in a maximum error of $10^{-3}\pi$. Increasing the transform length further only marginally reduces the error, while enhancing the intensity resolution beyond 10 bits can further improve accuracy. With a resolution of 12 bits and a transform length of at least $2^7 = 128$, the maximum phase error is reduced to $3 * 10^{-4}\pi$.

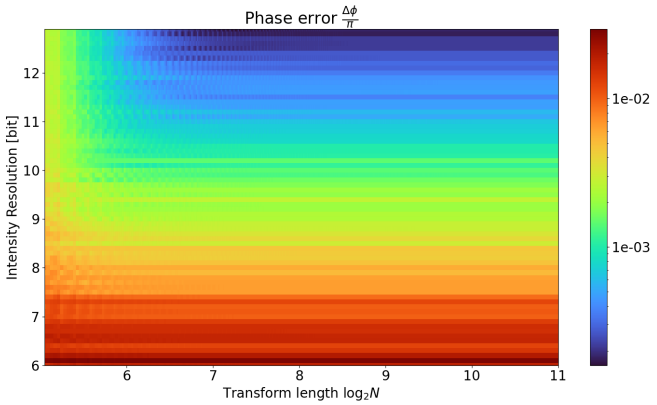

Figure 5.11: Phase error as a function of intensity resolution and transform length; color scale depicts the maximum relative phase error in units of π with Hanning window

For the results of the following sections it is important to be aware of the phase measurement accuracy of the current test setup. As explained before, the intensity of the optical lattice is equivalent to an effective intensity resolution of 6 bit. Additionally taking the camera noise and transform length of $2^9 = 512$ into consideration, a maximum phase error of $10^{-2}\pi$ is obtained (see Figures 5.5 and 5.11).

5.2 Phase Measurement

A typical phase signal has a drift of approximately 0.5π over the range of 5 s to 30 s with faster fluctuations in a range of 0.1π (see Figure 5.12). Decomposing the signal into its spectral components, it can be seen that the phase drift contains relevant frequency components up to 50 Hz to 100 Hz (see Figure 5.13).

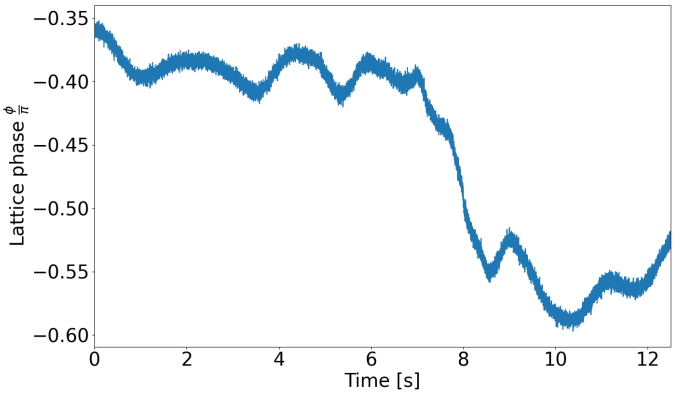

Figure 5.12: Lattice phase measurement over 12.5 s using FPGA

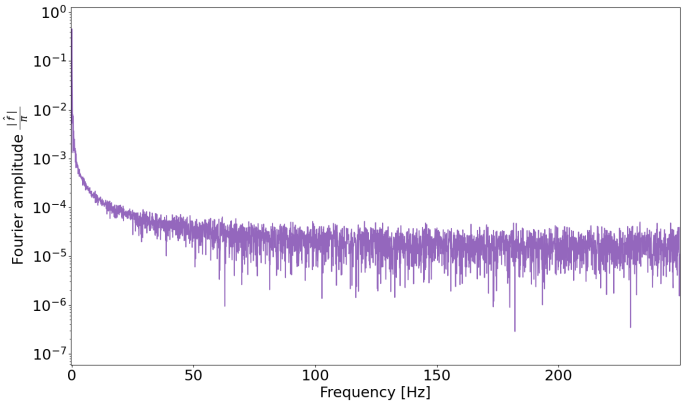

Figure 5.13: Frequency spectrum of the unstabilized lattice phase

By inspecting the phase measurement over a shorter period of time, as in Figure 5.14, it is observable that the value of the calculated phase differs between consecutive pictures by up to $8 * 10^{-3}\pi$. Taking into consideration the analysis in Section 5.1.5, where the accuracy of the phase measurement for the current setup was estimated to be $10^{-2}\pi$, it can be concluded that the sample-to-sample phase differences are not caused by a physical phase drift but are a result of the measurement error. These fluctuations are of course not expected to be stabilizable with a controller.

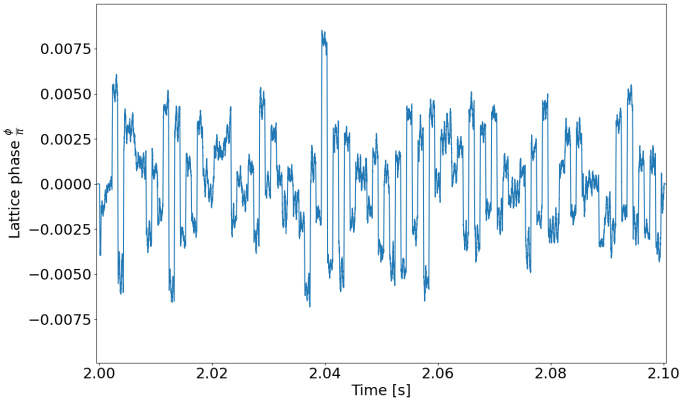

Figure 5.14: Lattice phase measurement over 0.1 s using FPGA; close-up of the measurement shown in Figure 5.12 between 2 s to 2.1 s; for easier identification of the fluctuation range, the signal is centered around zero; the sampling period of 1 ms is reflected in the step-wise change of the signal; signal fluctuations within these steps are caused by the DAC ouput noise and oscilloscope measurement

5.3 PI Control

5.3.1 PC-based Control

Figure 5.15 shows a direct comparison of two phase measurements taken with the PC, one with stabilization, the other without. The setpoint is set to zero constantly in this first case. It is evident that the higher frequency fluctuations cannot be stabilized using a PI controller with an update rate of 20 Hz. Nevertheless, the long-term drift of the phase can be removed with stabilization. This is underlined by the comparison of the two signals in frequency domain depicted in Figure 5.16. The frequency components up to approximately 1 Hz are shifted to higher frequencies.
These results show that the feedback stabilization works in principle, but it requires a significantly higher update rate than 20 Hz to stabilize the frequency components up to 50 Hz to 100 Hz.

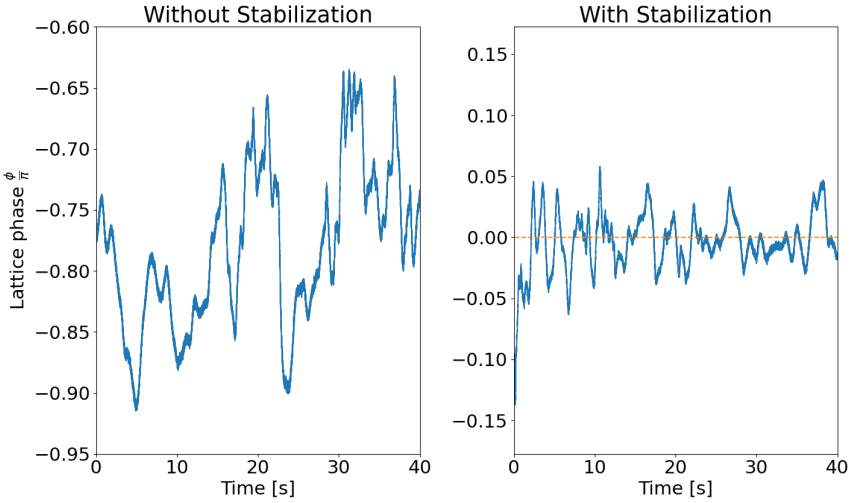

Figure 5.15: Comparison of unstabilized and stabilized phase signal with PC-based design; the setpoint is set to zero for the whole time range (orange dashed line); axis ranges have been adjusted to match for both signals

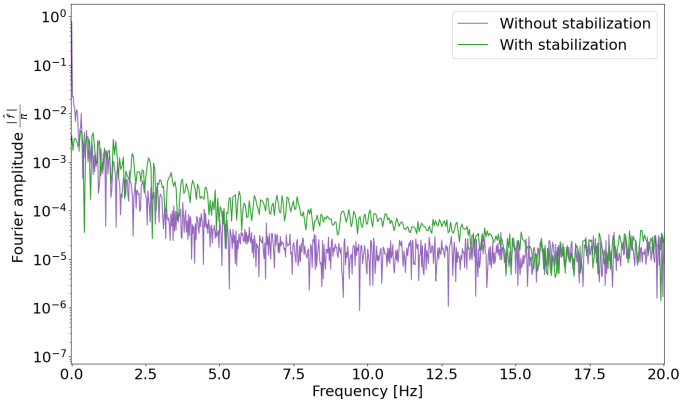

Figure 5.16: Comparison of unstabilized and stabilized phase signal in frequency domain with PC-based design

In a second test, a step input is given as setpoint. The step response of the control system is shown in Figure 5.17. The phase stabilizes around the new setpoint successfully, however, the transition is relatively slow. Because the signal fluctuations are faster than the feedback rate, the PI parameters have to be chosen conservatively to avoid oscillations. A side effect of this is low responsiveness of the controller to a change in setpoint.

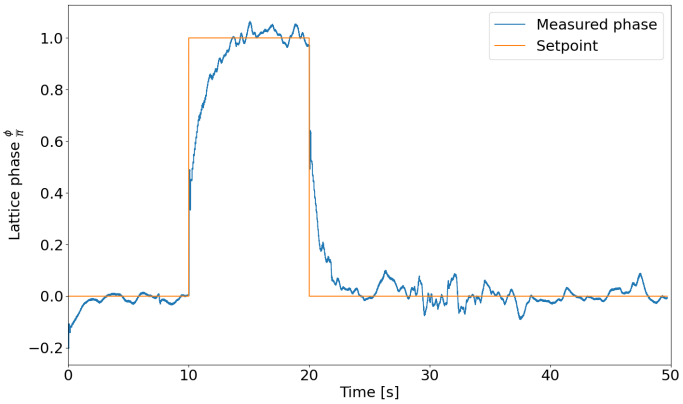

Figure 5.17: Step response with PC-based design

5.3.2 FPGA-based Control

The same tests are also performed with the FPGA calculating the output signal instead of the PC. Figure 5.18 displays the results of stabilization to a constant setpoint of zero. The lattice phase reaches the setpoint in 10 ms after the controller is activated. It stays within $\pm 10^{-2}\pi$ of the setpoint.
Comparing the stabilized signal with the unstabilized one in frequency domain yields the results shown in Figure 5.19. With the active stabilization the frequency components of up to 45 Hz can be reduced. To compare, the sample rate of the phase measurement is 1 kHz. The bandwidth of the controller can thus be estimated to be 45 Hz.

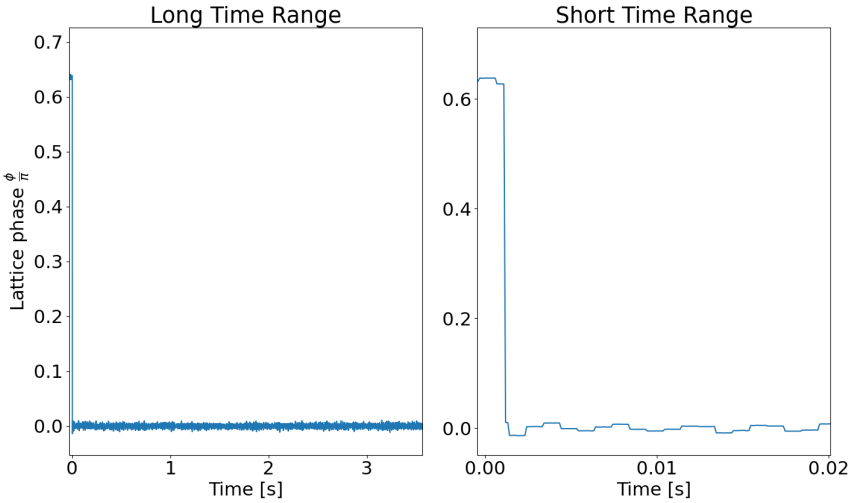

Figure 5.18: Stabilized signal with FPGA-based design; left: stabilization over the span of 3.5 s; right: zoom into the first 20 ms of the stabilization process

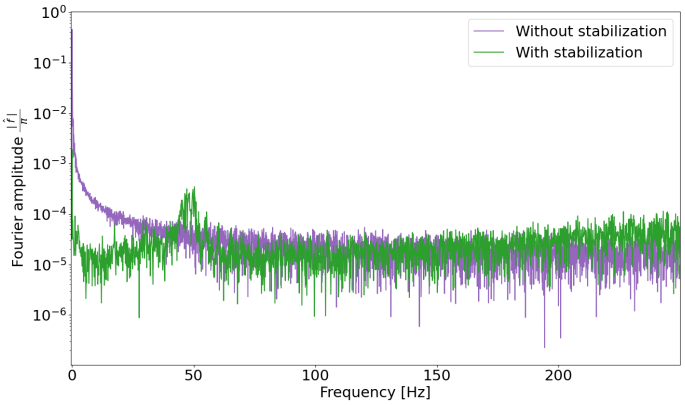

Figure 5.19: Comparison of unstabilized (purple) and stabilized (green) phase signal in frequency domain with FPGA-based design

Secondly, a step is again applied as a setpoint as seen in Figure 5.20. The lattice phase reaches the new setpoint within 10 ms and stays within a range of $\pm 10^{-2}\pi$ after that.

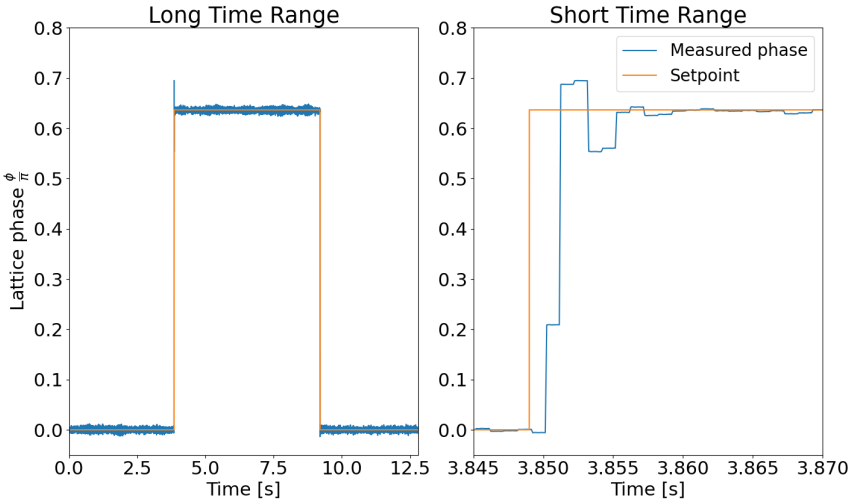

Figure 5.20: Step response with FPGA-based design

Further work is necessary to enable a change of setpoint over multiple lattice sites. This is difficult due to the jump of the phase output at $\pm\pi$ and the limited range of the function generator output. It was observed that changing the output of the function generator from 360° to 0° causes a significant jump in the lattice phase so simple rewinding at the range boundaries is not possible.

Chapter 6

Conclusion

The goal of this work was the implementation of a real-time phase stabilization of an optical lattice based on FFT and acousto-optic steering. The target platform chosen is an FPGA, which allows for a low-latency yet flexible implementation.

To gain more insight into the system, a prototype of the stabilization system was implemented on a PC. Testing demonstrated the feasibility of the phase stabilization approach but also underlined that a feedback rate of 20 Hz is not enough to control higher frequency lattice phase fluctuations.

The final FPGA design enables calculation of the lattice phase in 6.22 μs and uses only a small portion of the FPGA resources. Using this to realize a feedback rate of 1 kHz results in successful stabilization to within $\pm 10^{-2}\pi$ of the setpoint. The fluctuations of $\pm 10^{-2}\pi$ are a result of the limited accuracy of the phase measurement, and can thus not be removed using a PI controller. Apart from that, the active feedback allows the lattice phase to track the setpoint closely, with a settling time of 10 ms.

Additionally to the results of the control mechanism, the phase measurement error was analyzed in detail. Although additional tests are necessary to get definite results for the phase error on the FPGA, with the current setup phase measurements are expected to be accurate up to $\pm 10^{-2}\pi$. This could be improved by using a camera with better intensity resolution and signal-to-noise ratio. The accuracy heavily depends on the transform length or the dimensions of the image equivalently. It was demonstrated that transform lengths of at least $2^6 = 64$ pixels at an intensity resolution of at least 10 bits yield an accuracy of $\pm 10^{-3}\pi$. A higher intensity resolution, for example to 12 bits would lead to further improvement of the accuracy.

These considerations are important for future work on lattice phase stabilization. A priority for future work is the improvement of the phase measurement accuracy with a higher intensity resolution. Additionally, alternative cameras with a higher sampling rate will be investigated, possibly improving the bandwidth of the controller.
Moreover, one of the next steps is to extend the current approach to two-dimensional lattices including lattice structures consisting of multiple superimposed lattices. Firstly, this probably necessitates the incorporation of a second camera, which is already attainable with the existing design, requiring only minor adjustment. Secondly, processing not only one line of the image, but also the image's columns will most likely be necessary. As the transform length in vertical direction can only be increased at the cost of camera sample rate, it is important to consider the trade-off between transform length and measurement accuracy.
Additionally to the extension to two-dimensional lattices, the current design can be extended by using a function generator with a phase modulation range larger than 0° to 360°, which would enable rewinding at lattice site boundaries to control the position of the lattice over multiple sites. Furthermore, sending the images acquired on the FPGA would be beneficial for analysis and monitoring in the future, which could be achieved by extending the existing Ethernet interface.

Acknowledgements

Working on this bachelor's thesis has been a challenging yet extremely valuable experience, and I am deeply thankful to everyone who has played a part in its completion.
First of all, I would like to thank my supervisors, Marius Gächter, Samuel Jele and Kadir Akin, for composing this extremely interesting project, supporting me throughout and giving helpful advice. In addition, I would like to thank the whole Quantum Optics Group for creating such an enjoyable work environment and the interesting insights into research that I could learn a lot from. Last but not least, I would like to express my gratitude to Prof. Lukas Novotny, for supervising my project and providing valuable feedback even though my work was conducted outside the Photonics Laboratory.

Acronyms

AOD Acousto-Optic Deflector. 6, 12, 14

BRAM Block Random Access Memory. 10

CL Camera Link. 8, 9

DAC Digital-to-Analog Converter. 7, 10–13, 15, 20, 21

DFT Discrete Fourier Transform. 1, 3, 4, 23

FFT Fast Fourier Transform. 3, 6, 7, 13–15, 17–19, 22, 23, 26–28, 34, A-3

FIFO First In, First Out. 17, 18, 21, 22

FPGA Field-Programmable Gate Array. 1, 7–10, 12, 13, 22, 23, 26–28, 32, 34

GUI Graphical User Interface. 7, 15

IP core Intellectual Property core. 19, 20

LUT Look-Up Table. 10

MSB Most Significant Bit. 18

PL Programmable Logic. 7, 9, 10

PS Processing System. 10, 18, 20

SDK Software Development Kit. 14

SDP System Demonstration Platform. 7, 11, 15

SPI Serial Peripheral Interface. 10, 11, 15, 17, 21

Bibliography

[1] P. S. Jessen and I. H. Deutsch. *Optical Lattices - Advances in Atomic, Molecular, and Optical Physics, Vol. 37*. Academic Press, 1996.

[2] Florian Schäfer, Takeshi Fukuhara, Seiji Sugawa, Yosuke Takasu, and Yoshiro Takahashi. "Tools for quantum simulation with ultracold atoms in optical lattices". In: *Nature Reviews, Physics* (2020).

[3] Simon Wili, Tilman Esslinger, and Konrad Viebahn. "An accordion superlattice for controlling atom separation in optical potentials". In: *New Journal of Physics* (2023).

[4] Samuel Jele. *Engineering optical lattices in k-space*. 2023.

[5] K. M. M. Prabhu. *Window Functions and Their Applications in Signal Processing*. CRC Press Taylor Francis Group, 2014.

[6] M. Stefano, A. Akin, P. Fabritius, C. Studer, and T. Esslinger. *Real-time Laser Beam Profiling for Quantum Simulations with Ultracold Atoms*. 2021.

[7] *User Manual GO-5000M-PMCL*.

[8] Giacomo Bisson. *FPGA based fast camera readout and laser beam profiling*. 2021.

[9] *Mars XU3 SoC Module - Reference Design for Mars EB1 Base Board User Manual*.

[10] *Mars EB1 Base Board User Manual*.

[11] *Digilent PmodDA3 Digital To Analog Module Converter Board Reference Manual*.

[12] *Analog Devices: EVAL-AD5791SDZ User Guide*.

[13] *Analog Devices: SDP User Guide UG-277 - SDP-B Controller Board*.

[14] *PID Tuner application*. https://pidtuner.com. Accessed: 2023-10-04.

[15] Enis Mustafa. *Real-time Laser Beam Stabilization using FPGA*. 2022.

[16] *PG109 Fast Fourier Transform LogiCORE IP Product Guide*. https://docs.xilinx.com/r/en-US/pg109-xfft. Accessed: 2023-09-23.

[17] J. Schwider, R. Burow, K.-E. Elssner, J. Grzanna, R. Spolaczyk, and K. Merkel. "Digital wave-front measuring interferometry: some systematic error sources". In: *Applied Optics Vol.22* (1983).

[18] P. Hariharan, B. F. Oreb, and T. Eiju. "Digital phase-shifting interferometry: a simple error-compensating phase calculation algorithm". In: *Applied Optics Vol.26* (1987).

[19] Richard Leach. *Optical Measurement of Surface Topography*. Springer, 2011.

Appendix A

Alternative Phase Calculation Methods

A.1 Phase-Shifting Interferometry

In this work, an approach to stabilize the lattice phase using the FFT of the image was pursued. As the lattice's shape and spacing is known to a certain degree, the FFT approach does not make use of all the information available. An alternative method to calculate the lattice phase making use of the known shape of the interference pattern is presented in this chapter.

Instead of multiple lattice sites, only one period of the lattice is imaged with the camera. This results in an intensity distribution of the form

$$\begin{aligned} I(\Theta) &= A\cos^2\left(\frac{\Theta}{2}+\frac{\Delta\phi}{2}\right) \\ &= \frac{A}{2}+\frac{A}{2}\cos^2\left(\Theta+\Delta\phi\right) \\ &= \frac{A}{2}+\frac{A}{2}\left(\cos(\Theta)\cos(\Delta\phi)-\sin(\Theta)\sin(\Delta\phi)\right) \end{aligned}$$

with $\Theta = 0, \frac{2\pi}{N-1}, ..., 2\pi - \frac{2\pi}{N-1}, 2\pi$ and N is the number of data points of the image row.

Both $\sin(\Delta\phi)$ and $\cos(\Delta\phi)$ can be extracted from this intensity pattern with the appropriate choice of probing points Θ. An algorithm proposed by Schwider et al [17] and Hariharan et al.[18] utilizes 5 measurements of the intensity:

$$\sin(\Delta\phi) = 2\left(I\left(\frac{3\pi}{2}\right)-I\left(\frac{\pi}{2}\right)\right)$$

$$\cos(\Delta\phi) = 2I(\pi)-I(0)-I(2\pi)$$

$$\Delta\phi = \arctan\left(\frac{2(I(\frac{3\pi}{2})-I(\frac{\pi}{2}))}{2I(\pi)-I(0)-I(2\pi)}\right)$$

This method is based on phase-shifting interferometry, where an accurate measurement of the phase difference between two waves is of importance.[19]

The suitability for this application was investigated with simulations based on the same approach as in Section 5.1.1. Figure A.1 presents the accuracy of the calculated relative phase between two simulated images. The phase measurement error is displayed for different deviations from the expected lattice spacings. This influences the accuracy because this method relies on an accurate knowledge of the period of the lattice in order to probe the lattice at the right points. Additionally, it is assumed that the lattice spacing is a multiple of 4 (in pixels). The reason for this is that, the 5-point algorithm can use the value of the single pixels directly. In this case, the phase shifting method surpasses the FFT algorithm in accuracy.
If the lattice spacing is not a multiple of 4, the points at $I(\frac{\pi}{2}), I(\pi)$ and $I(\frac{3\pi}{2})$ cannot be probed directly.

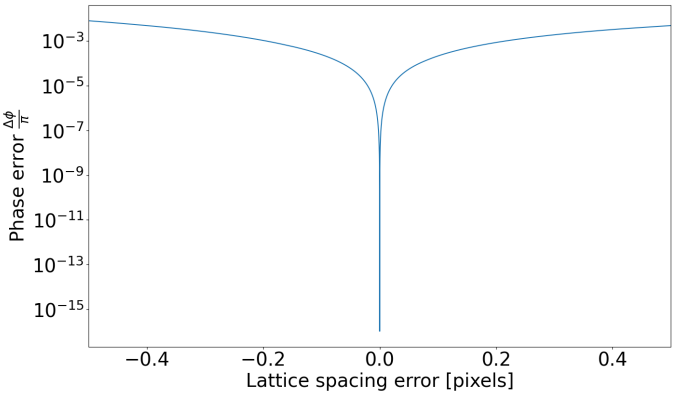

Figure A.1: Algorithmic maximum error of phase-shifting method for a range of deviations between real lattice spacing and expected lattice spacing; the expected lattice spacing is set to $d = 4$ pixels for this analysis

However, similar to the FFT phase calculation method, the phase shifting method's accuracy decreases if the finite intensity resolution of the data points is taken into account. Figures A.2 and A.3 display the algorithmic error for 6-bit and 12-bit intensity resolution, respectively.

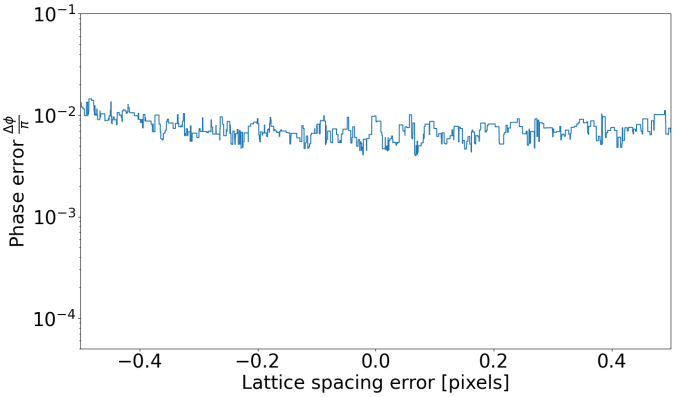

Figure A.2: Algorithmic maximum error of phase-shifting method for 6-bit intensity resolution and a range of deviations between real lattice spacing and expected lattice spacing; the expected lattice spacing is set to $d = 4$ pixels for this analysis

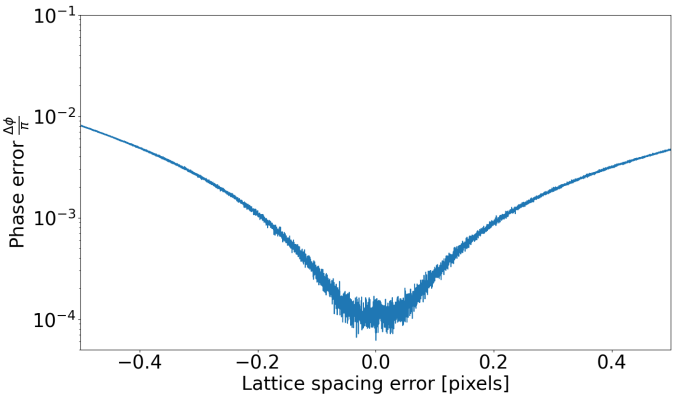

Figure A.3: Algorithmic maximum error of phase-shifting method for 12-bit intensity resolution and a range of deviations between real lattice spacing and expected lattice spacing; the expected lattice spacing is set to $d = 4$ pixels for this analysis

These results demonstrate that the error of this method is of similar order as for the FFT approach.

However, this approach comes with some additional challenges. In contrast to the FFT approach, the lattice spacing cannot be validated using this measurement technique, meaning its accuracy is dependent on the estimate of the lattice spacing.
Moreover, to adapt this method for general lattice spacings that aren't multiples of 4 pixels, additional techniques like weighted means or calculations of the smallest common multiple must be introduced. Using a weighted mean approach, the two pixels closest to the points $x = \frac{\pi}{2}, \pi$, and $\frac{3\pi}{2}$ can be employed to estimate $I(\frac{\pi}{2}), I(\pi)$, and $I(\frac{3\pi}{2})$, respectively. Alternatively, multiple lattice sites can be imaged, allowing the utilization of pixels at $x = \frac{\pi}{2} + 2n_1\pi, \pi + 2n_2\pi$, and $\frac{3\pi}{2} + 2n_3\pi$ (where $n_1, n_2, n_3 \in \mathbb{N}$) as probing points.
Furthermore, it remains uncertain whether the phase-shifting method can be extended as elegantly to more complex lattice structures as the FFT approach. One advantageous characteristic of phase measurement using FFT is that superimposed lattices with different lattice spacings show as distinct peaks in the Fourier transform, which allows separate calculation of their lattice phases. Separating such superimposed lattices using the phase-shifting method is more challenging. Because of this and the fact that the accuracy is not significantly higher, the phase-shifting-interferometry method was not pursued further.

A.2 Phase Overlap Scanning

Apart from the phase-shifting interferometry method described in the previous section, one could also imagine calculating the phase by comparing the overlap of the real intensity pattern over one period of the lattice with pre-generated signals of a specific phase.
For this, functions of the following form are stored:

$$I_i(x) = \cos^2(k_x x + \phi_i)$$

for N equally spaced phases $\phi = -\frac{\pi}{2}, -\frac{\pi}{2} + \frac{\pi}{N}, ..., \frac{\pi}{2} - \frac{\pi}{N}$ and $i = 0, 1, ..., N-1$.
The N functions can all be generated from the same data points by circular rotation, meaning only one table of values has to be stored effectively. In principle, multiple k_x can also be generated with the same table of values.
To determine the lattice phase, the overlap o_i with the acquired intensity pattern $I_{real}(x)$ is calculated for all N phase values as follows:

$$o_i = \sum_{j=0}^{R-1} I_{real}(j) * I_i(j)$$

where R is the number of pixels of the image line.
The resulting lattice phase is the ϕ_i with the largest overlap o_i. The accuracy of this algorithm is mainly limited by the number N of phase values probed, as it is accurate to approximately $\frac{1}{2N}$. However, a deviation between the expected and lattice spacing, or k_x equivalently, increases the algorithmic error.
Similar to the FFT approach, this method is also heavily limited by the finite intensity resolution of the camera. The measurement accuracy was determined in simulations. The results revealed maximum errors of $4 * 10^{-2}\pi$ for 8-bit and $1.5 * 10^{-2}\pi$ for 12-bit intensity resolution, with no further reduction achievable even with a larger N. It can be concluded that this method has a slightly lower maximum accuracy compared to the other two approaches, while also requiring a larger number of pixels. Hence, it was omitted in favor of the current FFT-based phase calculation.

Appendix B

FPGA Resource Utilization

Resource	CLB LUTs		CLB Registers		CARRY8		F7 Muxes		F8 Muxes	
Total available	70560		141120		8820		35280		17640	
axis spi dac	163	0.231%	74	0.052%	1	0.011%	0	0.000%	0	0.000%
async fifo phase	122	0.173%	165	0.117%	0	0.000%	0	0.000%	0	0.000%
async fifo pid	122	0.173%	165	0.117%	0	0.000%	0	0.000%	0	0.000%
Mars XU3 block design	8575	12.153%	9137	6.475%	72	0.816%	268	0.760%	37	0.210%
pmod3 spi	63	0.089%	97	0.069%	6	0.068%	0	0.000%	0	0.000%
image processing total	3419	4.846%	5820	4.124%	337	3.821%	0	0.000%	0	0.000%
arctan (cordic)	1430	2.027%	1479	1.048%	224	2.540%	0	0.000%	0	0.000%
fft input wrapper	13	0.018%	30	0.021%	0	0.000%	0	0.000%	0	0.000%
fft	1582	2.242%	3904	2.766%	84	0.952%	0	0.000%	0	0.000%
fifo	1	0.001%	1	0.001%	0	0.000%	0	0.000%	0	0.000%
fifo input wrapper	277	0.393%	83	0.059%	18	0.204%	0	0.000%	0	0.000%
find max	116	0.164%	302	0.214%	11	0.125%	0	0.000%	0	0.000%
pid controller	167	0.237%	193	0.137%	16	0.181%	0	0.000%	0	0.000%
xcounter	37	0.052%	35	0.025%	0	0.000%	0	0.000%	0	0.000%
ycounter	18	0.026%	14	0.010%	0	0.000%	0	0.000%	0	0.000%
TOTAL	12464	17.664%	15458	10.954%	416	4.717%	268	0.760%	37	0.210%

Resource	CLB		LUT as Logic		LUT as Memory		Block RAM		DSPs	
Total available	8820		70560		28800		216		360	
axis spi dac	60	0.680%	163	0.231%	0	0.000%	0	0.000%	0	0.011%
async fifo phase	40	0.454%	122	0.173%	0	0.000%	1.5	0.694%	0	0.000%
async fifo pid	30	0.340%	122	0.173%	0	0.000%	1.5	0.694%	0	0.000%
Mars XU3 block diagram	1941	22.007%	6235	8.836%	2340	8.125%	2	0.926%	0	0.816%
pmod3 spi	19	0.215%	63	0.089%	0	0.000%	0	0.000%	0	0.068%
image processing total	885	10.034%	2734	3.875%	685	2.378%	4.5	2.083%	58	3.821%
arctan (cordic)	260	2.948%	1424	2.018%	6	0.021%	0	0.000%	0	2.540%
fft input wrapper	6	0.068%	13	0.018%	0	0.000%	0	0.000%	0	0.000%
fft	572	6.485%	904	1.281%	678	2.354%	2	0.926%	54	0.952%
fifo	1	0.011%	1	0.001%	0	0.000%	1	0.463%	0	0.000%
fifo data wrapper	53	0.601%	277	0.393%	0	0.000%	1.5	0.694%	0	0.204%
find max	55	0.624%	115	0.163%	1	0.003%	0	0.000%	4	0.125%
pid controller	48	0.544%	167	0.237%	0	0.000%	0	0.000%	2	0.181%
xcounter	9	0.102%	37	0.052%	0	0.000%	0	0.000%	0	0.000%
ycounter	5	0.057%	18	0.026%	0	0.000%	0	0.000%	0	0.000%
TOTAL	2975	33.730%	9439	13.377%	3025	10.503%	9.5	4.398%	58	4.716%

Table B.1: Resource utilization on the FPGA; configurable Logic Blocks (CLBs), Look-Up Tables (LUTs), Multiplexers (Muxes), Block RAM, Digital Signal Processing (DSP) blocks utilization in absolute units and percentage of available resources; split into the various submodules of the design; Mars XU3 block design includes the clock generation and the Ethernet interface, and the resources would thus not have to be doubled to accomodate a second image pocessing module

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

FPGA-Based Phase Stabilization of an Optical Lattice

Verfasst von (in Druckschrift):

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Schnuck	Maria Sophie

Ich bestätige mit meiner Unterschrift:

- Ich habe keine im Merkblatt „Zitier-Knigge" beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Zürich 13.10.2023	M. Schnuck

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.