A High-Speed 40-channel USB 3.0 DAC for Adaptive Optics

Laurens van Dam
Sander van Dijk
Michel van der Kaay

Delft University of Technology

TU Delft

This document was typeset by LaTeX.

A High-Speed 40-channel USB 3.0 DAC for Adaptive Optics

by

Laurens van Dam
Sander van Dijk
Michel van der Kaay

to obtain the degree of Bachelor of Science
at Delft University of Technology,
to be defended on June 30, 2015 at 03:30 PM.

ISBN 978-94-6186-494-9

L. (Laurens) van Dam	4203321
S.A. (Sander) van Dijk	4227018
M.S. (Michel) van der Kaay	4240804

Date of submission:	June 22, 2015	
Project duration:	April 20, 2015 – July 3, 2015	
Supervisor:	Prof.dr. G. Vdovin	
Thesis committee:	Prof.dr. K.L.M. Bertels,	TU Delft, Jury Chairman
	Dr.ir. J.A. Martinez-Castaneda,	TU Delft, Jury Member
	Dr. V. Patlan,	OKO Flexible Optical B.V., Supervisor

TU Delft Delft University of Technology

Preface

As a conclusion of our three year BSc programme in Electrical Engineering there is the Bachelor Graduation Project. This thesis is the final result of this project conducted at Delft University of Technology, faculty Electrical Engineering, Mathematics and Computer Science. The goal of this project is to prove the academic skills of the authors and test the capability of designing an electronic system in a structured manner. Officially, the project period was the 20th of April until the 3rd of July.

The supervisor and commissioner of the project is Prof.dr. G. Vdovin, full professor at Delft Center for Systems and Control (DCSC). The project is commissioned by a company founded by the supervisor, Flexible Optical B.V., which is specialized in the field of research and application-oriented development of laser and high resolution imaging adaptive optics.

The thesis focuses specifically on Systems Engineering, where a Digital-to-Analog Converter (DAC) control system is designed that has a higher refresh rate than the current DAC systems on the market. This enables more modern instruments that operate at higher frequencies to be incorporated with a DAC that supports these higher frequencies. Its main purpose is to be used in a feedback loop coupled with possible measurement systems, which can be used in all kinds of applications. The project was split into two parts for two different implementations: one that is compatible with USB 2.0, and one with USB 3.0 (SuperSpeed) for an even higher refresh rate. The latter implementation (with USB 3.0 capability) is described in this thesis.

The result of this research is a (proof-of-concept) prototype that can be integrated in a measurement setup. The measurement setup is then enhanced with the capabilities of a USB 3.0 (SuperSpeed) Digital-to-Analog converter.

Laurens van Dam
Sander van Dijk
Michel van der Kaay

Delft, June 2015

Acknowledgements

We would like to thank the supervisor of our project, Prof.dr. G. Vdovin, for commissioning the assignment and guiding us through the process. Next we cannot forget Dr. Seva Patlan, Senior Researcher at Flexible Optical B.V., to be thanked. We thank him for conducting several meetings with us on a weekly basis, and taking the time to answer any questions we had. Our thanks go out to Ad van Dijk from ADAS Automatisering for helping us with implementing the prototype and giving us feedback on the preliminary version of the thesis. Dr.ir. Arjan van Genderen and dr.ir. Michiel Pertijs took the time to review our thesis, for which they receive our thanks. Furthermore we would like to thank our other team members from the USB 2.0 implementation, Yvo Mulder, Joost Romijn and Danilo Verhaert for the support and positive mindset during the period of the project, and outside of it. We would like to thank Arrow Electronics, Inc. for letting us borrow a Cypress FX3 Evaluation Board. Concludingly, our thanks go out to TransIP for supplying us with a virtual server for any collaboration tools and storage needs.

Abstract

A High-Speed Digital-to-Analog converter system is proposed that is controlled using USB 3.0. Possibilities to improve the refresh rate and resolution are researched to meet the standards of modern measurement systems. For instance, adaptive optics often uses a DAC in its feedback loop, but the USB DAC systems currently available provide insufficient refresh rates to support modern measurement systems. To aim for a higher resolution, the feasibility of applying audio DACs in the system is researched. And to control the DACs, a hardware-based subsystem of the MCU receives USB data through DMA and interfaces with the DACs. Furthermore, a prototype is built and tested. This prototype is capable of updating the DAC outputs synchronously with a rate of 20 kHz.

Contents

1

Introduction

The radiation coming from our surroundings is hard to observe. We are dealing with all kinds of deviations or *aberrations* in the atmosphere. In optics an aberration is a deviation of an image compared to the ideal optical image. For example when an optical system is not correctly focused, a blurred image occurs. These aberrations arise due to impurities in the media that the electromagnetic waves need to radiate through, for example due to atmospheric turbulence or thermal effects. When trying to observe these waves on a small scale, the aberrations cause an image to become hard to analyze. In order to obtain a clear image the aberrations need to be corrected for.

This is where the technology called *adaptive optics* makes a difference. It is a technology that improves the performance of an optical system by correcting the deviations or disturbances of a wavefront. An example of such an optical system could be a measurement setup that performs continuous measurements of a laser beam. To measure the aberrations in the wavefront and compensate for them, one could use a deformable mirror that constantly adapts to the aberrations of the resulting wavefront. To construct this system one is inherently bound to a system that has a high speed at which it can observe these aberrations. In the system there is the need of a sensor that observes the current wavefront. Because the sensor has a digital output, a control system can be developed that will in turn control this deformable mirror in such a way that the wavefront will stay free of aberrations. This is where a DAC plays a role in converting the output of the control system to a suitable input for the mirror. Its feedback system plays an ever increasing role in improving system performance. The better performance of the DAC, the better performance of the system possibly can be. See Figure 1.1 for a schematic overview of this example measurement setup where adaptive optics is applied.

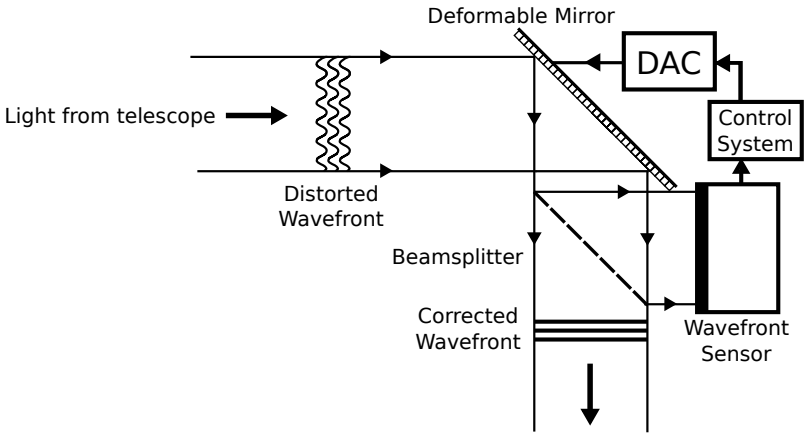

Figure 1.1: Example of a setup where the High-Speed DAC could be used in the field of Adaptive Optics. Light from a source contains a distorted wavefront that can be corrected through a deformable mirror. In order to feed the control system with the current wavefront, a beamsplitter is used. The DAC then converts the output from the control system to the deformable mirror.

This thesis focuses on developing the digital-to-analog converter (DAC) system that is fast enough to comply with the current optical technologies on the market. Therefore, one of the made choices is to make use of a USB 3.0 SuperSpeed connection. The project is a result of a proposal commissioned by Flexible Optical B.V. in the Netherlands. Flexible Optical B.V. is a small company specialized in the field of application-oriented development of laser and high resolution imaging using adaptive optics. The 'next generation' USB DAC system, which is designed during this project, will have improved specifications over the already existing DAC system developed by Flexible Optical B.V. The main improvement compared to the existing DAC system is an increased refresh rate and resolution.

The main contents of this thesis is the design and implementation process. Any specific choices that are made during these processes are elaborated. The thesis will focus on the USB 3.0 implementation of the DAC system and describes a proof of concept to bring the design in practice. In Chapter 2 the problem definition is discussed. An overview of the complete DAC system and its components is described in Chapter 3. Any background information related to the system and its components is explained in Chapter 4. Chapter 5 focuses on the implementation of the DAC system. The prototype is then evaluated and the results are documented in Chapter 6, where any discussions regarding these results are documented as well. Any ethical matter concerning the subject of this thesis resides in Chapter 7. Concludingly, the final conclusions with recommendations are described in Chapter 8.

2

Problem Definition

In order to meet the requirements of current technology, the required specifications of a DAC in the field of adaptive optics have changed. This demands a faster and more precise DAC system. The previous DAC system from Flexible Optical B.V. was developed almost a decade ago, and technology has improved since then. The main purpose of this DAC is to control deformable mirrors used in adaptive optics. Adaptive optics is a technology to correct optical imperfections in real-time, as mentioned in the introduction. Most measurement setups for adaptive optics contain a feedback system, in which a DAC is used to convert the signals from the control system to an analog signal for the deformable mirrors.

The new DAC system implemented in this project is expected to have an increased refresh rate of 20 kHz instead of 1 kHz. This means that all output channels of the DAC can be updated simultaneously 20.000 times per second. The reason of increasing this refresh rate is that the DAC should support the typical frequencies of the processes that produce disturbances. For example, for atmospheric turbulences the typical frequency is believed to be in the range of 100 Hz to 10 kHz, depending on the conditions. Introducing a faster DAC device would potentially allow for an expansion of applicability of the DAC.

Another improvement for the new DAC system is that the required resolution is increased from 12 bits to 16 bits per channel. Increasing the resolution results in a greater dynamic range and a higher level of precision to control the deformable mirrors, which is always desirable.

These are the most important factors of improvement for the DAC system that is to be implemented. Applications of adaptive optics are nowadays not limited to astronomy only. For example, it is also used in biological imaging and in many other fields of research. Regardless of its application, a higher refresh rate for the new DAC system is always desirable. Table 2.1 contains an overview of the current specifications of the DAC system and the 'next-generation' DAC system.

Table 2.1: Existing and 'next generation' DAC specifications.

Parameter	Existing DAC system	'Next generation' DAC system
Analog outputs	40 channels	40 channels
Refresh rate	1,000 frames per second	20,000 frames per second
Resolution	12 bits per channel	16 bits per channel
Mode	Synchronous for all channels	Synchronous for all channels
Output range	0 V - 5 V with adjustable limits	0 V - 5 V or −5 V - 5 V with adjustable limits
Power	Supplied by USB	Supplied by USB
Max. load capacitance	> 500 pF	> 500 pF

3

Overview of the System and its Components

3.1. System Overview

This chapter explains the proposed *Digital-to-Analog Converter* (DAC) system and the overview of system components. The system consists of a USB controller, a *Microcontroller Unit* (MCU), 40 DACs and a power regulator, which is illustrated in Figure 3.1. Both data and power are provided via USB. The MCU processes the data and sends it to the DACs. The DACs then provide analog outputs, which are transformed to the correct bipolar output range using amplifiers. Refer to Figure 5.5 for a detailed overview of how the DACs are connected to the MCU.

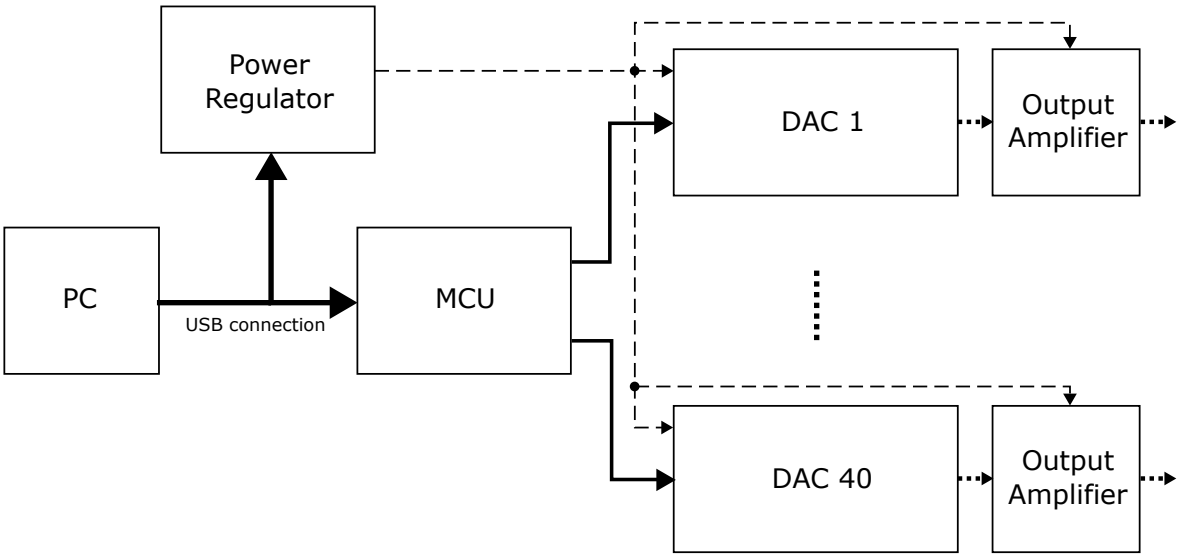

Figure 3.1: Global overview of the DAC system. The MCU controls every individual DAC. An output amplifier follows after every DAC output. Power is distributed through the power regulator, which gets its main power from USB. Dashed lines represent power lines. Dotted lines are analog output signals of the DAC and the Output Amplifiers.

3.2. Component Overview

The main components of the DAC system are briefly introduced. Specific information on the actual implementation can be found in Chapter 5.

3.2.1. Universal Serial Bus (USB) Controller

A USB controller is required to facilitate the USB connection between the host (a PC) and the MCU. A USB 3.0 connection is chosen to achieve higher refresh rates and lower latencies that were not possible with earlier versions of USB. This USB 3.0 controller can either be implemented into the MCU or exist as a separate bridge in the system. Microcontrollers and bridges implementing USB 3.0 are however still uncommon since USB 3.0 is still not widely adapted in the industry.

3.2.2. Microcontroller (MCU)

The microcontroller is the component that processes the incoming USB data and reformats it to suit the DACs' data interface. Also, DAC configuration may be performed by the MCU, with either data from USB or with pre-defined settings.

3.2.3. Digital-to-Analog Converter (DAC)

The DAC is the component that converts the digital data from the MCU into analog output signals. The DAC should drive 40 analog outputs. This can be implemented with a single 40-channel DAC IC or multiple DAC ICs with less channels.

3.2.4. Output Amplifier

The output amplifier buffers and amplifies the analog outputs from the DAC to the correct range. To drive a load capacitance of $500\,\mathrm{pF}$, which is a design requirement, buffering is required since DACs are usually unable to do so. Therefore, buffering is always required, either in the DAC IC itself or using an external buffer.

When the DAC itself does not output the correct voltage range, this can be amplified and shifted using the output amplifier as well.

3.2.5. Power Regulator

A power regulator is needed in the system to supply the required voltage levels for the microcontroller, the DAC and the output amplifier. A system requirement is that the system is USB powered, meaning that power should be supplied by the single USB connection.

4

Background on System Components

Before the implementation and the design choices are explained, the necessary theory needs to be explained. A general system overview of a DAC system is shown in Figure 4.1. This section describes background theory on these components that is necessary to make design choices for the implementation. First, basic information about digital-to-analog converter techniques that could be interesting for this implementation are discussed. Next, the multiple ways of interfacing with the DAC ICs are discussed. Furthermore, the Universal Serial Bus protocol is evaluated and a comparison is made between USB SuperSpeed and its previous versions. Finally, the theory behind a possible output amplifier is discussed and a method to estimate the effective number of bits of a DAC system is proposed.

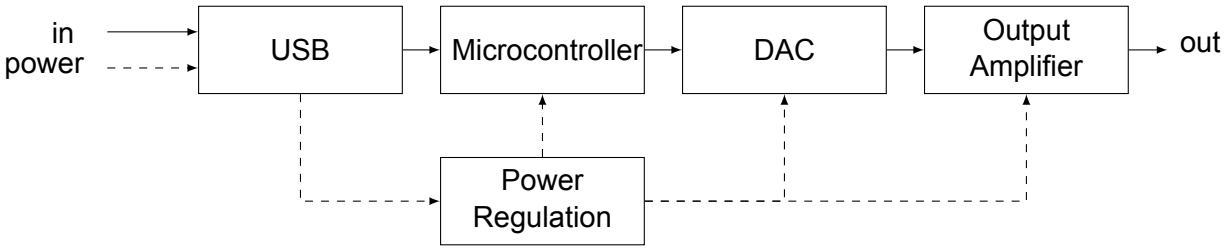

Figure 4.1: Block diagram of the USB DAC. Dashed lines represent power lines, solid lines represent signals.

4.1. Possible Digital-to-Analog Converters

The Digital-to-Analog converter converts the digital input from USB into an analog output. This section will review different types of DACs.

4.1.1. Resistive DAC

The first type of DAC is the traditional *resistive DAC*. This is the most simple implementation of a DAC and was therefore first on the market. The most widely used resistive DACs are now discussed, which are the R2R DAC, Binary Weighted DAC and the R-String DAC.

R2R DAC

The R2R DAC is the first variant of resistive DACs. It consists of two different resistor values throughout the circuit as the name suggests: R and 2R, in which the absolute value of the resistors do not matter in theory, as long as the second of the two types of resistors is twice as high as the first resistor [1]. Figure 4.2 illustrates the R2R circuit with n input bits. a_0 to a_{n-1} illustrate the individual bits from the digital input. Those voltage values are either the reference voltage connected to ground, depending on whether a_i with $0 \leq i \leq n-1$ is 0_2 or 1_2.

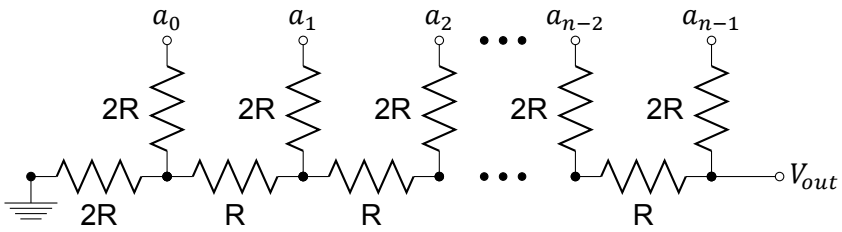

Figure 4.2: Schematic overview of an R2R DAC.

The main problem of resistive DACs is their accuracy at higher bit resolutions. For instance, for a 16-bit R2R DAC, one of the resistors of the MSB should have a maximum error of

$$\frac{\Delta R}{R} < \frac{1}{2^{16\text{ bits}}} \approx 0.0015\%. \tag{4.1}$$

Therefore, resistors with an accuracy of 0.0015% are required to obtain the precision of 16 bits. These accurate resistors are extremely expensive compared to conventional resistors. An error plot of a 16-bit R2R DAC can be found in Figure 4.3. The output voltage corresponding with a single LSB here is

$$V_{LSB} = \frac{V_{ref}}{2^{bits}} = \frac{5\text{ V}}{2^{16}} \approx 76\ \mu\text{V}, \tag{4.2}$$

therefore the resolution is approximately $76\,\mu$V.

Figure 4.3 contains a plot of the error for a 16 bit R2R DAC with 1% accuracy tolerance resistors and a unipolar output span of 1 V. The plot was made using a MATLAB simulation in which random deviations from the nominal values were created within the resistor tolerance of 1%. This results in the maximum error of the output of 10 mV, which is way higher than the resolution. This error occurs at half of the reference voltage where the MSB switches on and the rest of the bits off, having the largest impact on the error since the resistors of the MSB influence the output voltage more than the lower significant bits.

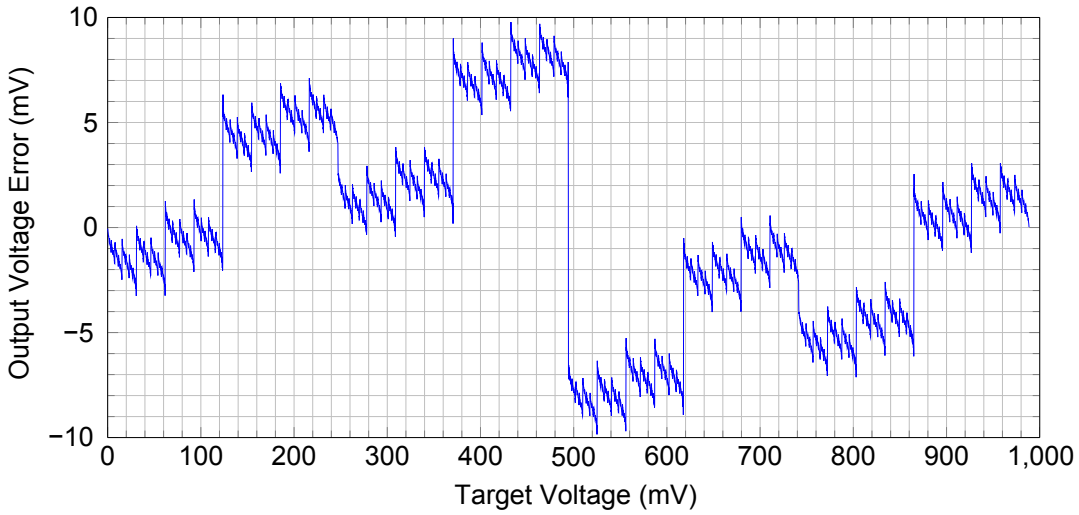

Figure 4.3: 16-bit R2R DAC output voltage error plot with 1% resistors (1V unipolar voltage reference).

Resistors in the R2R network closer the the MSB have a greater impact than the resistors closer to the LSB. Therefore for resistors close to the MSB it is more important to have an accurate resistor value. The closer a resistor is to the LSB, the less the impact of the resistor accuracy on the final analog output value is. This also allows for lower accuracy resistors to be used in positions closer to the LSB.

It is difficult to implement high resolution R2R DACs due to the high amount of high precision resistors needed. This is often accomplished on a digital integrated circuit where resistor values can be precisely trimmed.

When a pair of resistors is lower than the targeted value, a correction can be applied using the lower significant bits to correct the somewhat lower output voltage to the exact voltage. However, when the resistor is higher than the targeted value, correction is not always possible. When in the case of 1000000000000000_2 the MSB resistor values are higher than expected, the corresponding output voltage will be greater than half of the reference voltage. This makes the output voltage incorrect because the MSB would create a higher voltage than the expected voltage. It might be possible to lower this voltage difference by lowering the digital input value by 1 LSB. This change would ideally make a difference of 1 LSB on the output. The difference between 1000000000000000_2 and 0111111111111111_2 can be more than 1 LSB if the resistors are not accurate. This creates an error that can not be corrected.

Binary Weighted DAC

The binary weighted DAC uses the ability to add binary weighted currents using an operational amplifier [2]. The total current is then converted to the output voltage. The structure of a N-bit binary weighted DAC is shown in Figure 4.4.

The binary weighted DAC requires only one resistor per bit, but the values should be doubled per bit, requiring a large variety of resistor values. Since double values of resistors are not that commonly available in practice, it might be necessary to increase resistors by a factor of two by combining other values in series or parallel in order to obtain the right resistor values. ICs are not bound to this problem since they can implement any resistance with high precision.

The thermal noise of the topology increases with the resistor values [4]. Therefore, low value resistors should be used. However in this DAC a large variety of resistors is needed, thus both low and high value resistors are used. Consequently, when the amount of bits is increased the noise is also increased. This is not desirable when aiming for a high resolution DAC.

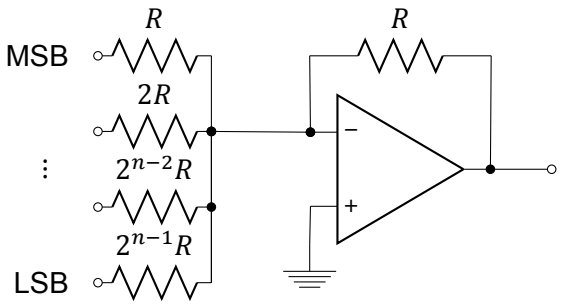

Figure 4.4: An N-bit binary weighted DAC.

For the binary-weighted DAC it is important to have high precision resistors especially for the lower valued resistors. Those resistors will have the most influence on the output voltage (and thus are connected to bits close to the MSB). With this type of DAC it is not difficult to implement a bipolar output range by applying a negative voltage to the bit. At the R2R DAC there is the problem that only one direction of correction is possible. This is not the case for the binary weighted DAC because it is bipolar per bit. Namely, correction can be done by either switching a bit high, neutral or negative.

R-String DAC

An R-string DAC is a kind of DAC that creates an analog output voltage based on a big voltage divider. The targeted voltage is selected with switches [3]. The R-string DAC requires a large amount of switches compared to the R2R and Binary Weighted DAC. Figure 4.5 shows a schematic representation of a 3-bit R-String DAC.

Compared to other kinds of resistive DACs, an R-string DAC contains a relatively high amount of components. An R-string DAC contains 2^N resistors, where N is the amount of bits. Because of the high amount of resistors and switches necessary, this type of DAC is generally used in integrated circuits because integrated circuits can easily contain large amounts of switches and resistors. Besides, the resistors can be trimmed to obtain near exact values preventing the need to use external, expensive high-accuracy resistors. In an R-string DAC all the resistors have an equal impact on the output voltage. Therefore it is not possible to compensate for inaccurate resistors in the R-string. However the impact of an inaccurate resistor is smaller than 1 LSB since 2^N resistors are used in the R-string.

A large advantage of the R-string DAC is that it is guaranteed to be monotonic. This means that an increased value at the digital input can not result in a decreased analog output value, which is the case for the R2R and binary weighted DAC. Monotonicity prevents the possibility that feedback systems get stuck since a new equilibrium is created where both increasing and decreasing the digital input result in negative feedback. Since the proposed DAC system is used in a feedback loop, monotonicity is required.

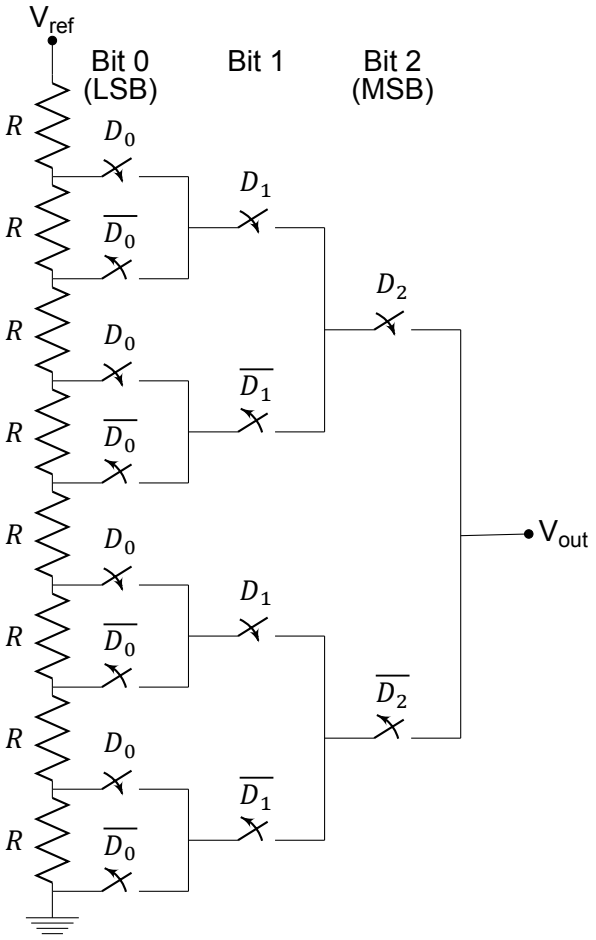

Figure 4.5: A 3-bit R-String DAC.

4.1.2. Pulse-Width Modulated DAC

Pulse-Width Modulation (PWM) is a form of signal modulation whereby the data is represented in square waves with equal amplitudes but different average amplitudes. The frequency of this square wave is constant, but the ON time in a period may vary. The ratio between the ON time and the total period is also known as the duty cycle. The square wave with the duty cycle is then low-pass filtered to get the average value of the square wave, which is the analog output [6].

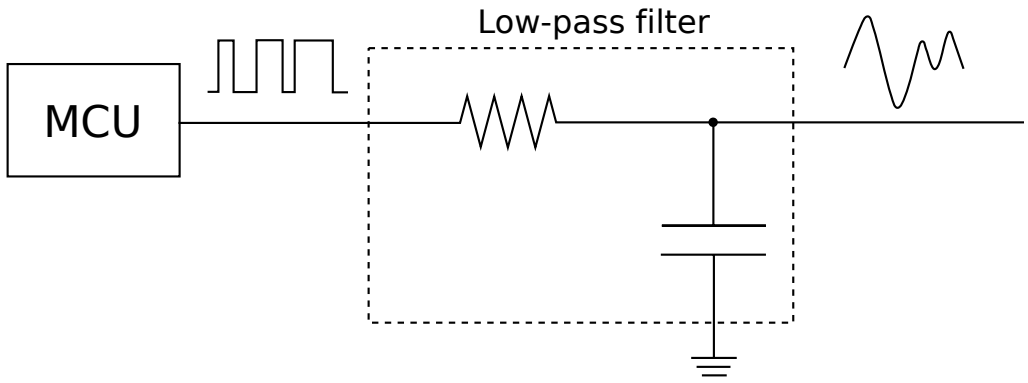

Figure 4.6: Principle of operation of a PWM DAC with a simple low-pass filter consisting of a resistor and a capacitor. An example input and output signal is shown.

An advantage of a PWM DAC is that it is relatively easy to implement since only an MCU and low-pass filter is needed. A constraint is that the voltage on the output of an MCU should be stable and precise. However, the output voltage of the MCU is directly proportional to the analog output value [5].

High precision is hard to achieve on a PWM DAC since processor clock speeds determine the amount of time steps in a period. Also, the low-pass filter does not allow a quick change in output. Therefore, high speed DACs are hard to implement using PWM DACs.

As for the R-string DAC, the PWM DAC is monotonic by design, which is desirable when used in a feedback loop.

4.1.3. Delta-Sigma DAC

A Delta-Sigma DAC is a kind of DAC that makes use of Delta-sigma modulation and a low-pass filter. Delta-Sigma modulation is commonly used in integrated circuits for their low cost and high accuracy. The Delta-Sigma modulator encodes a digital signal into a signal with a higher sample rate, but lower resolution. Subsequently, the high frequency signal goes to a low pass filter. In the low-pass filter the signal is transformed into an analog signal and unwanted high-frequency signals are removed [7]. A disadvantage of the low-pass filter is that even high frequency components that are desired, for example square waves, are suppressed. Figure 4.7 contains a block diagram of a Delta-Sigma DAC.

Same as the PWM DAC, the delta-sigma DAC is monotonic by design. An increase in digital input value can not result in a decrease of output value.

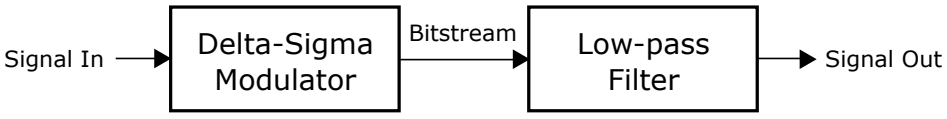

Figure 4.7: Schematic Delta-Sigma DAC setup.

The Delta-Sigma modulator will convert the digital signal into a bitstream [8]. Figure 4.8 shows the contents of this first block.

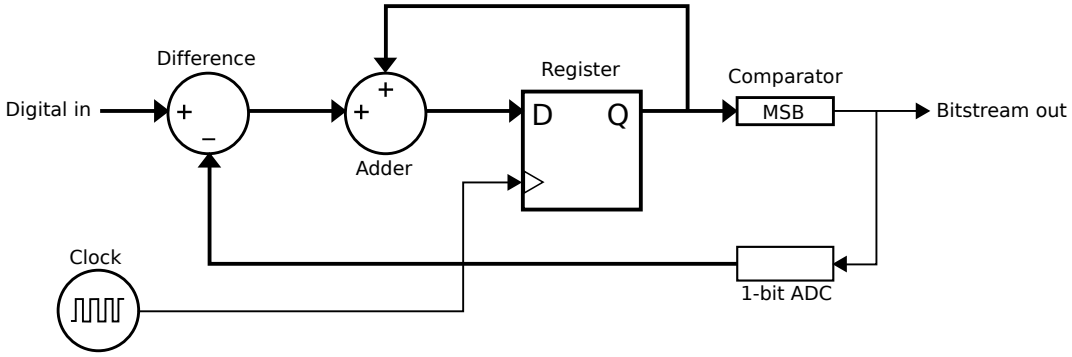

Figure 4.8: Block Diagram of a basic digital Delta-Sigma modulator [7].

This bit stream is basically a one-bit serial signal with a bit rate that is much higher than the data rate of the DAC itself. This one bit only represents two different values. In the unipolar case for example it represents $0\,\mathrm{V}$ and $5\,\mathrm{V}$ and in the bipolar case $-5\,\mathrm{V}$ and $5\,\mathrm{V}$. The average level of this bit stream then represents the input signal level. This bit stream can be compared with the bit stream of a PWM signal, discussed in Section 4.1.2. Lastly, this signal goes through a low-pass filter since all the information of the bit stream itself is in the lower frequency band and the low-pass filter ensures that frequencies above are filtered.

4.1.4. Comparison of DAC Types

Based on the previous sections, a comparison between the different kinds of DACs in terms of advantages and disadvantages can be made. This comparison is shown in Table 4.1.

Table 4.1: Advantages and disadvantages of different kind of DACs.

DAC Type	Advantages	Disadvantages
R2R	Little hardware needed	Not possible to correct for inaccurate resistors Not monotonic by design
Binary weighted	Little hardware needed Bipolar output span possible Correction for inaccurate resistors possible	Large variety of resistors needed Relatively much thermal noise Not monotonic by design
R-String	High accuracy possible Monotonic by design	Much hardware needed
Pulse-Width Modulation	Simple implementation Monotonic by design	Analog output voltage inaccurate High frequencies suppressed by low-pass filter
Delta-Sigma	High resolution and accuracy possible Monotonic by design	High frequencies suppressed by low-pass filter

4.2. DAC interfaces

There are several protocols that can be used in order to communicate with a DAC packaged in an IC. The most common protocols for transferring the data that is to be converted are *Serial Peripheral Interface* (SPI) for DACs in general and *Integrated Interchip Sound* (I^2S) in particular for audio DACs.

4.2.1. Serial Peripheral Interface (SPI)

SPI is a synchronous serial communication interface which interfaces between a master and a slave. SPI has a high throughput and is quite flexible. The speed is not limited. For example, bitrates of $50\,\mathrm{Mbit\,s^{-1}}$ are possible. SPI is often used to communicate with DAC chips either for data transmission or to set control signals [9]. SPI uses a master-slave construction, with one master. A *Slave Select* (SS) line selects the desired slave to transfer to. Furthermore, the *Serial Clock* (SCLK) line provides the clock rate from master to its slaves. The *Master Output Slave Input* (MOSI) line and *Master Input Slave Output* (MISO) lines are meant to transfer the data between both parties bi-directionally.

4.2.2. Integrated Interchip Sound (I^2S)

I^2S is a serial protocol that inherits quite some features from SPI. The main difference is that I^2S features a fixed sample rate. This implies that I^2S is meant for continuous transmissions which is desirable in audio applications.

I^2S is meant to send data for two different channels after each other on one line. There is a Left/Right Clock (LRCK) signal that clocks the data for the left channel when LRCK is low, and data for the right channel when LRCK is high. All of the implementations use at least three signals on the bus: the serial clock, a left/right select clock and the serial data line itself [10]. The left/right clock (LRCK) is used to switch between two channels (e.g. left and right channel).

There are multiple variations of I^2S that function in a similar way, but with some small differences. Two of these variations are *Left-Justified* and *Right-Justified*.

In the case of I^2S, then the data is meant for the right channel when LRCK is high. However, for Left-Justified and Right-Justified it is the opposite: the data is meant for the right channel when LRCK is low. Furthermore, for I^2S the MSB of the data is sent one SCLK cycle after the LRCK falling edge. This is in contrary with Left-Justified where the MSB of the data is sent directly on the rising edge of LRCK. Right-Justified is characterized by sending the LSB on the falling edge of LRCK, making the complete data bit sequence being justified to the right.

To clarify these differences between I^2S, Left-Justified and Right-Justified, timing diagrams are shown in Figures 4.9, 4.10 and 4.11 for respectively I^2S, Left-Justified and Right-Justified.

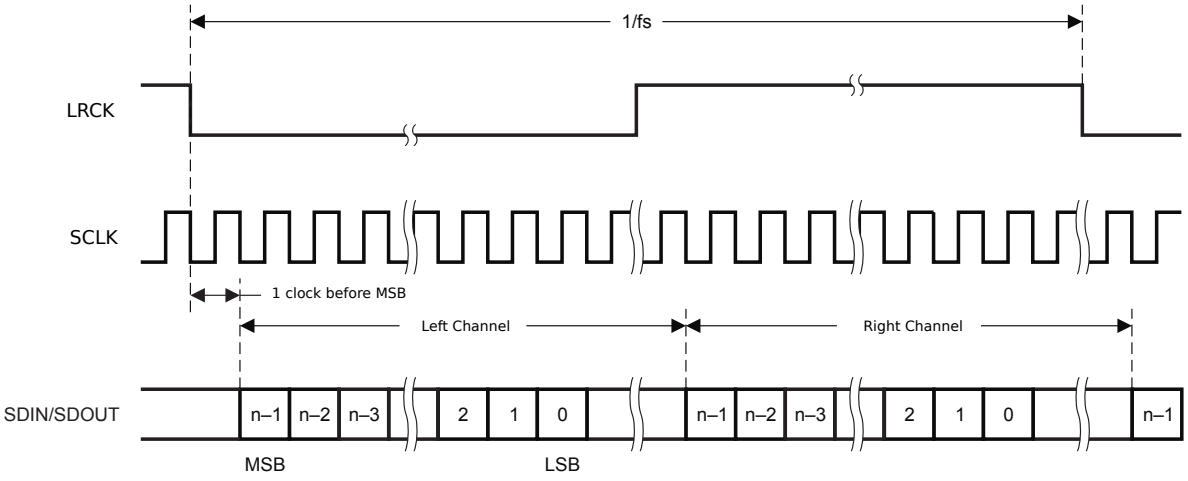

Figure 4.9: I^2S timing diagram [11].

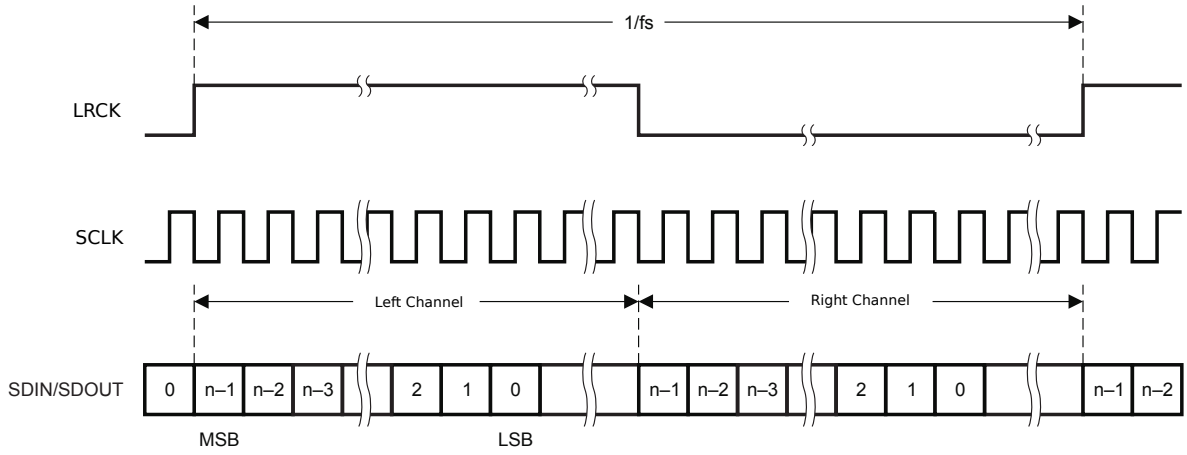

Figure 4.10: Left-Justified timing diagram [11].

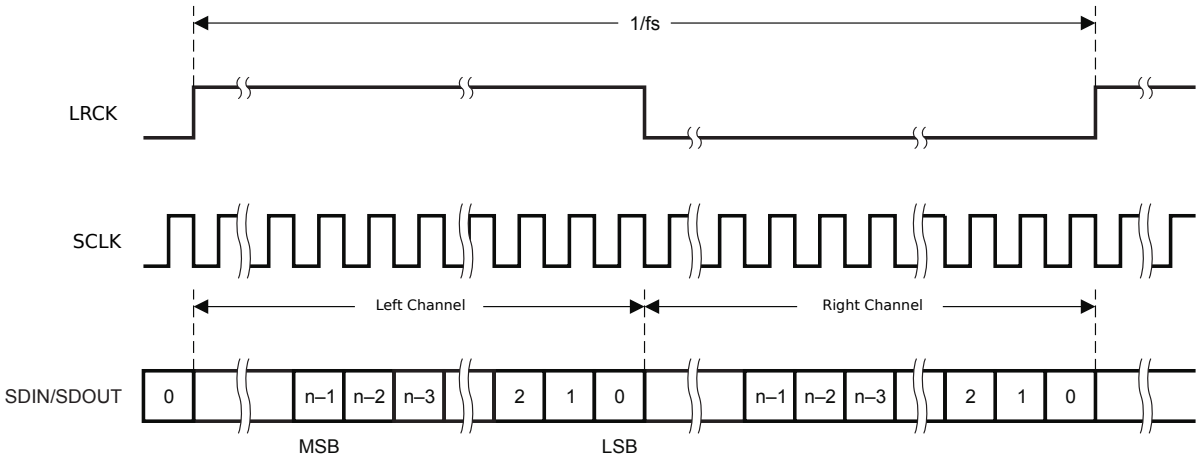

Figure 4.11: Right-Justified timing diagram [11].

4.3. Investigation of Universal Serial Bus 2.0 and 3.0

USB is the required interface between the host and the DAC system. This host could be any device supporting the USB protocol, but in most cases it would be a desktop computer. USB was mainly designed for simplicity, especially plug-and-play and compatibility, rather than high performance [12]. However, with USB 3.0 it is possible to achieve higher performance compared to its predecessor, USB 2.0. Using USB 3.0 would then result in both the ease of plug-and-play and a high transfer rate.

In this section, the ways of transmitting data through USB is discussed. This includes an estimation of the maximum achievable refresh rate through USB and the different transmission modes of USB. Furthermore, a comparison is made between USB 3.0 and USB 2.0, and the backward compatibility of USB 3.0 is discussed.

4.3.1. Estimation of Maximum Achievable Refresh Rate

In USB 2.0, the host receives and transmits data in fixed timeslots called *frames* or *microframes* for respectively the low speed and both the high speed and SuperSpeed variant [13]. This interval determines the amount of time between two DAC data frames, that is the period between two sequential packets that contain the values for all channels of the DAC. The frequency corresponding to this interval is called the *refresh rate*, which is fixed per USB variant: 1 kHz for USB 1.0 Low Speed and USB 2.0 Full Speed, and 8 kHz for USB 2.0 High Speed.

For USB 3.0 SuperSpeed mode (excluding USB 2.0 compatibility mode), the principle of refresh rate has been changed. Instead of waiting for the next timeslot, asynchronous notifications are used to indicate a request to send data, from both device and host. It is therefore not possible to define a fixed refresh rate for USB 3.0.

4.3.2. USB Transmission Modes and their Characteristics

USB distinguishes four different transmission modes [14], each optimized for a different purpose. All transmissions take place through a so-called *pipe* which can only support only one transmission mode and direction, except for control mode which is bidirectional. This means that when multiple transmission modes and/or directions are required, multiple pipes are required as well. Characteristics per transmission mode are listed in Table 4.2 [15].

Table 4.2: USB 2.0 transfer mode characteristics [14].

Transfer type		Control	Bulk	Interrupt	Isochronous
Typical use		configuration	printer, scanner	mouse, keyboard	audio
Required		yes	no	no	no
Allowed on low-speed devices		yes	no	yes	no
Data types/millisecond per transfer, maximum possible per pipe. Assumes data/transfer = maximum packet size [kb/s]	High speed	15,872 (31x64-byte transactions/ microframe)	53,248 (13x512-byte transactions/ microframe)	24,576 (3x1024-byte transactions/ microframe)	24,576 (3x1024-byte transactions/ microframe)
	Full speed	832 (13x64-byte transactions/ microframe)	1,216 (19x64-byte transactions/ microframe)	64 (1x64-byte transactions/ microframe)	1,023 (1x1023-byte transactions/ microframe)
	Low speed	24 (3x8-byte transactions/ microframe)	not allowed	0.8 (8 bytes per 10 ms)	not allowed
Direction of data flow		in and out	in or out	in or out	in or out
Reserved bandwidth for all transfers of the type [kb/s]		10 at low/full speed, 20 at high speed (minimum)	none	90 at low/full speed, 80 at high speed (isochronous & interrupt combined) (maximum)	
Error correction		yes	yes	yes	no
Message or stream data		message	stream	stream	stream
Guaranteed delivery rate		no	no	no	yes
Guaranteed latency (maximum time between transfers)		no	no	yes	yes

4.3.3. Differences between USB 3.0 and USB 2.0

In Table 4.3 the main differences between USB 2.0 and USB 3.0 are listed. Some of the differences are then explained. Regarding backwards compatibility between both versions, refer to Section 4.3.4.

Table 4.3: USB 2.0 vs USB 3.0 [16].

	USB 2.0	USB 3.0
Data Rate	$480\,\mathrm{Mbit\,s^{-1}}$ (High Speed) $12\,\mathrm{Mbit\,s^{-1}}$ (Full Speed) $1.5\,\mathrm{Mbit\,s^{-1}}$ (Low Speed)	$5.0\,\mathrm{Gbit\,s^{-1}}$ (SuperSpeed)
Data Interface	Half-duplex Two-wire differential signaling	Dual-simplex Four-wire differential signaling
Signals	Four signals: - 2x USB 2.0 data (D+, D-) - 2x Power (VBUS and GND)	Nine signals: - 4x SuperSpeed data (SSRX+, SSRX-, SSTX+, SSTX-) - 2x USB 2.0 data (D+, D-) - 3x Power (VBUS and 2x GND)
Bus transaction protocol	Host directed Polled traffic flow Packets broadcast to all downstream devices No multiplexing of data streams	Host directed Asynchronous notifications Packets routed only to target device Multiple data streams possible for Bulk transfers
Power management	Two modes: - Active - Suspend	Four modes: - Active (U0) - Idle, Fast (U1) - Idle, Slow (U2) - Suspend, Slow (U3)
Bus Current	Low-Power: 100 mA High-Power: 500 mA	Low-Power: 150 mA High-Power: 900 mA
Port state	Port hardware detects connect events. System software uses port commands to transition the port into an enabled state.	Port hardware detects connect events and brings the port into operational state ready for SuperSpeed data communication.
Transmission modes†	Control Bulk Interrupt Isochronous	(SuperSpeed) Bulk (SuperSpeed) Interrupt (SuperSpeed) Isochronous Bulk Streaming

†Changes in transmission modes for USB 3.0

All transmission modes are still present in USB 3.0, but with some changes [17]. All of the transmission modes allow bidirectional use in a single pipe, unlike USB 2.0 where only the control mode allowed bidirectional transmission. Also, bidirectional transfers can now occur in parallel since one differential data pair per transmission direction is now available instead of the single pair for both directions.

A new feature in USB 3.0 is *bulk streaming*. It supports multiple bulk transfers in parallel using a single bulk pipe, which was not possible in USB 2.0. Bulk streaming is also not bound to timeframes anymore, enabling higher refresh rates. The USB device is notified of the queued messages through the control pipe on forehand, decreasing the overhead that was required between each message separately. This enables a stream of packets with minimal overhead [18].

The required bandwidth of bulk transfers is efficiently allocated by filling up the gaps of unused bandwidth. Approximately 10% of the available USB 3.0 bandwidth is allocated to control transmissions, and up to 90% for periodic transfers like interrupts or isochronous transfers. The bandwidth that is not used can be used for bulk transfers [19]. This means that bulk transfers have a low priority which could be a disadvantage when using bulk transfers to control a DAC system, especially when there are multiple devices connected to the same USB controller on the host side. This would cause unwanted interrupts by other devices that could intervene with the low priority bulk transfer of the MCU.

Isochronous transmission mode in USB 3.0 still uses microframes, but can now be adjusted up to $\pm 13.33\ \mu s$ by sending configuration messages [20].

4.3.4. USB 3.0 Backward Compatibility

USB 3.0 implements two busses: the original USB 2.0 and the *SuperSpeed* extension [21]. The original two USB 2.0 signal wires exist alongside the extra four wires for SuperSpeed. The four extra wires are not connected in USB 2.0 female connectors, making it possible to use USB 3.0 as well as USB 2.0 in the original USB 2.0 female connector.

4.4. Output Amplifier

Depending on the type of DAC that is used, an output amplifier and filter is needed. For example a Delta-Sigma DAC or a PWM DAC needs an output filter as described in Sections 4.1.2 and 4.1.3. These kind of DACs have high frequency components that are undesired. A simple analog low-pass filter on the output of the DAC with a cutoff frequency on the Nyquist frequency would solve this [22]. However, most modern DAC ICs have an output filter integrated [1]. Besides an output filter, an output buffer is needed to buffer the analog signal.

Another function of an output buffer is to obtain the desired output span. According to the required specifications the output span should be unipolar from 0 to $5\,\mathrm{V}$ or bipolar from $-5\,\mathrm{V}$ to $5\,\mathrm{V}$. However, not all DACs have this specific output span by default. In that case an amplifier is needed. In Figure 4.12 an op amp configuration is shown a greater output span can be obtained. The output of the DAC is connected to V_{in}. A constant reference voltage is needed for an offset. In general, the output voltage of this configuration is in the form of [23]:

$$V_{out} = \text{gain} \cdot V_{in} + \text{offset}. \tag{4.3}$$

Using Kirchhoff's law an expression for the output can be obtained. This expression is,

$$V_{out} = V_{in} \cdot \frac{R_2}{R_1 + R_2}\left(1 + \frac{R_4}{R_3}\right) - V_{ref} \cdot \frac{R_4}{R_3}. \tag{4.4}$$

By choosing the right values of the resistors a precise gain and offset results. Any arbitrary output span can be reached with this configuration.

The output impedance of the DAC will influence the transfer function of the output amplifier. Therefore the values of the resistors R_1 and R_2 should be much higher than the output impedance of the DAC and the reference voltage supply. On the other hand, a higher resistor value will result in more noise. Since this is a voltage to voltage amplifier topology, a higher resistor value results in more noise on the output [4].

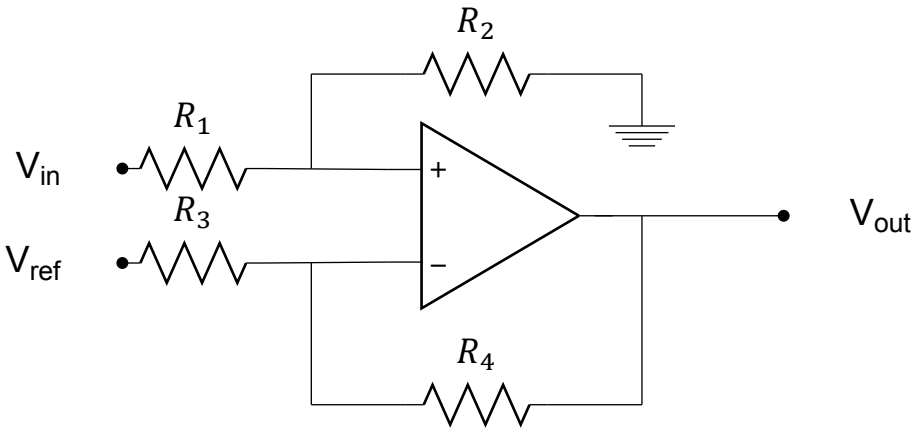

Figure 4.12: Output amplifier schematic with resistor values yet to be determined.

4.5. Estimation of the Effective Number of Bits

DAC performance influences the accuracy and the speed of the overall system. A DAC should meet specific requirements depending on the application they are used for. Therefore testing and evaluating DACs is essential for evaluating and comparing devices. For example noise can make DACs appear to have a smaller resolution than they actually have. A quantitative way of describing the number of effective bits is the *Effective Number of Bits* (ENOB) [24]. The ENOB specifies how many bits an ideal DAC would require to obtain the same performance as the real DAC. IEEE std.1241 defines the ENOB as ”a measure of the signal-to-noise and distortion ratio used to compare actual analog-to-digital converter (ADC) performance to an ideal ADC” [25]. In the case of ADCs an analog value is the input and a quantized digital representation is the output. For DACs both the digital input as the analog representation on the output are quantized. Still a similar definition for ENOB is true for DACs. The following expression is a measure of the ENOB expressed in ratios between the rms noise and quantization noise:

$$\text{ENOB} = N - \log_2\left(\frac{\text{rms noise}}{\text{ideal rms quantization noise}}\right) = \log_2\left(\frac{\text{full scale range}}{\text{rms noise } \cdot \sqrt{12}}\right) \text{ [bit]}, \tag{4.5}$$

where N is the number of digitized bits [24]. An alternative way of defining the ENOB, which can be derived in terms of the *Signal-to-Noise Ratio* (SNR), is:

$$\text{ENOB} = \frac{\text{SNR} - 1.76\,\text{dB}}{6.02} \tag{4.6}$$

Distortion of the analog output signal also influences the quality of the output signal. Therefore the ENOB is influenced by the distortion. An expression like (4.5) and (4.6) also is true for the distortion. A best measure for the effective number of bits is obtained when both nonidealities are taken into account. The *Signal-to-Noise and Distortion Ratio* (SINAD) is defined as [26]:

$$\text{SINAD} = 20\log_{10}\left(\frac{\text{S}}{\text{N} + \text{D}}\right) \text{ [dB]}, \tag{4.7}$$

where S is the signal power, N the noise power and D the distortion power. A new expression for ENOB is:

$$\text{ENOB} = \frac{\text{SINAD} - 1.76\,\text{dB}}{6.02} \tag{4.8}$$

5

System Implementation

This chapter explains the actual implementation of the DAC system. Any design choices made are discussed as well.

5.1. Cypress Microcontroller Evaluation Board

Since controllers and bridges that support USB 3.0 are still uncommon, only one suitable microcontroller was found. This is the Cypress CYUSB3031, a USB SuperSpeed (3.0) controller that features a 32-bit ARM926EJ processor which runs on 200 MHz [30]. The MCU also contains a *GPIF II* subsystem which is a flexible interface that can be used to create the interface between the microcontroller and the DACs. The used MCU is implemented on an evaluation board (FX3 Evaluation Board, CYUSB3KIT-001) which will be used for prototyping. This board contains the MCU and all necessary components like connectors and voltage regulators.

5.1.1. GPIF II

General Programmable Interface II is a flexible interface for the Cypress EZ-USB FX3 USB 3.0 Device Controller. This interface can be configured to suit most USB applications, either by using a supplied interface or by creating a new one from scratch [27]. The interface is designed using GPIF II Designer by Cypress.

GPIF II was introduced to achieve maximum sustained throughput in USB designs. GPIF II is capable of connecting the USB and the output interface faster than the processor could [28]. This is done by implementing this flexible interface entirely separately from the ARM processor, releasing it from handling the interface since the USB can be connected to the GPIF II directly. It is also possible to keep the processor in between, which gives the advantage of flexible preprocessing but the disadvantage of lower throughput.

The GPIF II subsystem on the MCU consists of a state machine, DMA subsystem and a port to both the ARM processor and the interface output, see Figure 5.1. This port, the P-Port, handles all incoming and outgoing signals:

- A bidirectional data/address bus which can be up to 32 bits wide. This is used to transfer the data from/to the DMA subsystem, potentially with address indications on this bus as well. To provide the addressing, the bus can be either time-multiplexed or split into separate data and address part.
- Up to 13 data control signals which can be used to implement additional signals that are used to correctly interpret the data from the data/address bus. For example, SPI's slave-select signal. These can be in, out or bidirectional.
- Flow control signals, which can be used to indicate the DMA buffer status to the processor for flow control. Seperate flags are available for when the buffer is full or filled to a certain programmable level, which is called a watermark.

The state machine is designed in the GPIF II Designer using a graphical user interface. States, actions to be performed in states and state transition conditions (events) can be combined into the total state machine to contain up to 256 states. The allowed actions are:

- Load data counter. Setup the start and overflow value, which are always the same everywhere in the state machine.
- Load address counter. Same as for the data counter but can contain a different configuration than the data counter.
- Count data. Increase the data counter by the step defined during the data counter load. When the overflow value is reached, an event is triggered on which the state machine can act. If the reload option is selected in the data counter configuration, the counter is reset to the start value.
- Count address. Same as for the data counter.
- Drive GPIO. Set GPIO to the active value or toggle it, depending on the GPIO configuration. When assert-mode (instead of toggle-mode) is selected in this configuration, the active state can be set to either HIGH or LOW. When no assertion is done in a state machine cycle, the assert-mode GPIO's are de-asserted to the idle state.
- Input data. Reads data from the data bus to the specified destination, which can be either DMA or the processor.
- Drive data. Write to the data bus from a DMA buffer or directly from the processor.
- Compare data. Compare data from the bus with the specified data. The event DATA_CMP_MATCH is triggered when the comparison matches, on which the state machine can transition to another state.
- Input address. The read value is used to select the DMA channel.
- Interrupt CPU. Send an interrupt to the processor which can see the state that sent the interrupt, but no further data can be send with the interrupt.
- Commit. Force the data packet to the DMA channel consumer.

The internal clock of 400 MHz is prescaled to the clock frequency at which the state machine will function. This prescaler can be any value between 2 and 1024, resulting in frequencies between 200 MHz and 390 kHz, respectively. It is possible to make this clock an output in order to create synchronous interfaces.

Every rising edge of the main clock, the actions of the current state are performed and a transition to another state is possible. It can be disabled to repeat the actions when no state transition is performed. The available events for transitioning to another state are:

- Unconditionally transition.
- Data counter overflow. Triggered when *Count data* overflowed the data counter in the previous cycle.
- Address counter overflow. Triggered when *Count address* overflowed the address counter in the previous cycle.
- DMA ready. Triggered when the selected DMA channel's buffer is full or empty, depending on the data direction.
- DMA watermark. Triggered when the selected DMA channel's buffer is above/below a certain, predefined amount of bytes in the buffer.

These events can be conditioned and combined using the AND, OR and NOT operators. When having multiple outgoing transitions from a state the conditions may not conflict. DMA events require to be exclusive to all transitions from a state, for example only *DMA channel 1 ready* and *NOT DMA channel 1 ready* are permitted as two possible transitions from a single state.

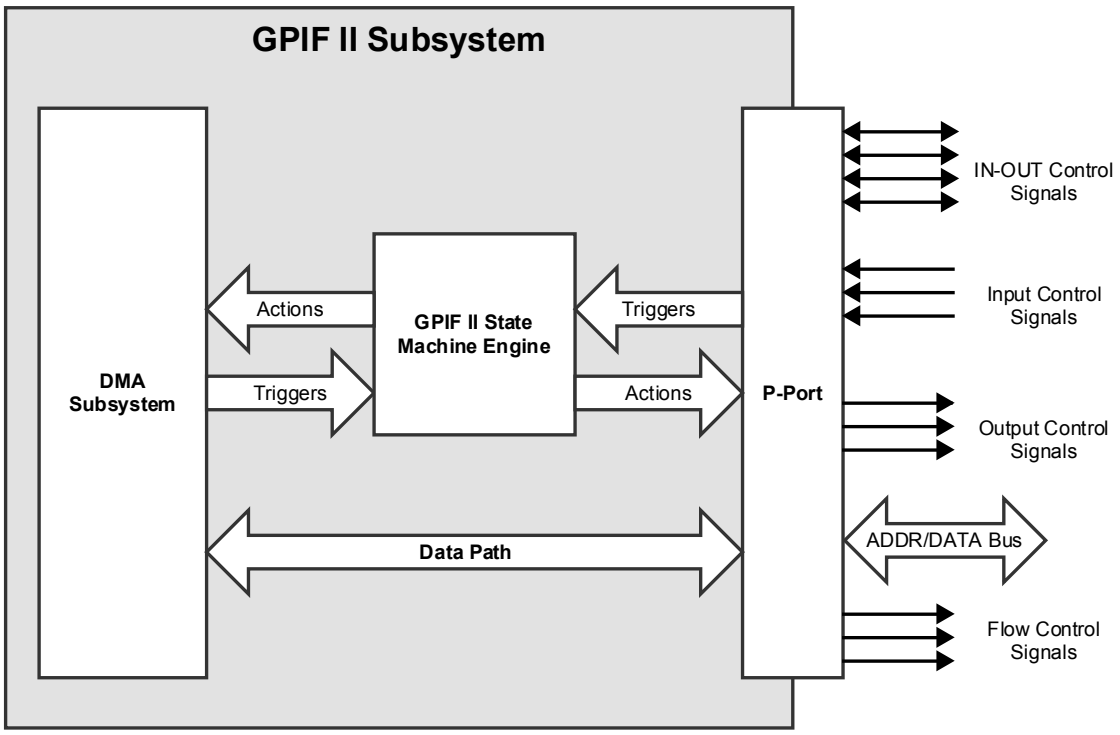

Figure 5.1: GPIF II subsystem on the CYUSB303x [29].

5.2. Digital-to-Analog Converter and its Features

In order to implement a high-speed DAC system, a high-speed DAC is needed. A consideration needs to be made on what type of DAC will be used. Refer to Section 4.1.4 for advantages and disadvantages of a couple of DAC techniques.

With a Delta-Sigma DAC it is relatively easy to achieve a higher resolution compared to resistive DACs. This is because resistive DACs are influenced by the precision of its resistors, or need a high quantity of resistors. Please refer to Section 4.1.1 for more detail on resistive DACs, and Section 4.1.3 for Delta-Sigma. Furthermore, with Delta-Sigma DACs it is possible to create sufficiently high sample rates to be suitable for our purposes. Because of these reasons, the choice has been made to make use of a Delta-Sigma DAC.

The kind of DACs that usually implement Delta-Sigma are DACs meant for audio applications. The choice has been made to use the Cirrus Logic CS4384. This DAC advertised as audio DAC, but due to this application it is possible to achieve high sample rates and a high resolution. This could significantly improve the performance of existing DACs systems on both refresh rate and resolution. An additional advantage is the low cost of audio DACs. The CS4384 is an 8-channel DAC, with a documented resolution of 24 bit which is achieved using Delta-Sigma modulation [31]. The serial data input on the DACs is a protocol similar to I^2S, which should be generated by the MCU. For more information on I^2S and its variations, refer to Section 4.2.2.

The resolution of the chosen CS4384 DAC is well over the required resolution of 16 bit, but a more precise DAC could prove more useful for future purposes, again for adaptive optics or perhaps in other applications. Furthermore this DAC is chosen for its high maximum sample rate of $216\,\mathrm{kHz}$. Each pair of DAC channels has its own serial inputs, making it possible to achieve this high sample rate. In combination with USB 3.0 it is possible to achieve higher sample rates than with existing USB DACs.

In order to achieve a 40-channel DAC system, one can place five CS4384 ICs in parallel in order to achieve $5 \times 8 = 40$ different output channels.

5.2.1. Data Input

The DAC features a serial input interface. The supported protocols are I^2S, Left-Justified, Right-Justified, *One-Line Mode* (OLM) and *Time-Division Multiplexing* (TDM) [31]. We use Left-Justified as transmission protocol since this was the least difficult to implement in the firmware of the MCU, and does not have any disadvantages compared to other variations of the I^2S protocol. For further information on Left-Justified, refer to Figure 4.10 in Section 4.2.2.

5.2.2. DAC Filters

For audio enthusiasts, the DAC contains a configurable interpolation filter. For our purposes we would like to keep the output unfiltered. The filter can be set to either a slow roll-off or a fast roll-off. Interpolation means that the signal is oversampled and thus smoothed [32]. The filter response plot of the rejection band for the fast and slow roll-off is shown in respectively Figure 5.2 and 5.3. The figures show that frequencies above 0.5 F_s, where F_s is the sample rate and equals the LRCK from Left-Justified, are suppressed for both the slow and fast roll-off filters. For the application of the DAC system in adaptive optics this filter is not desirable, since not only sinusoidal waves are desired on the output. These filters can not be disabled. For our application there is no difference between the slow and fast roll-off filter for the overall performance. Therefore, the default fast roll-off filter is chosen.

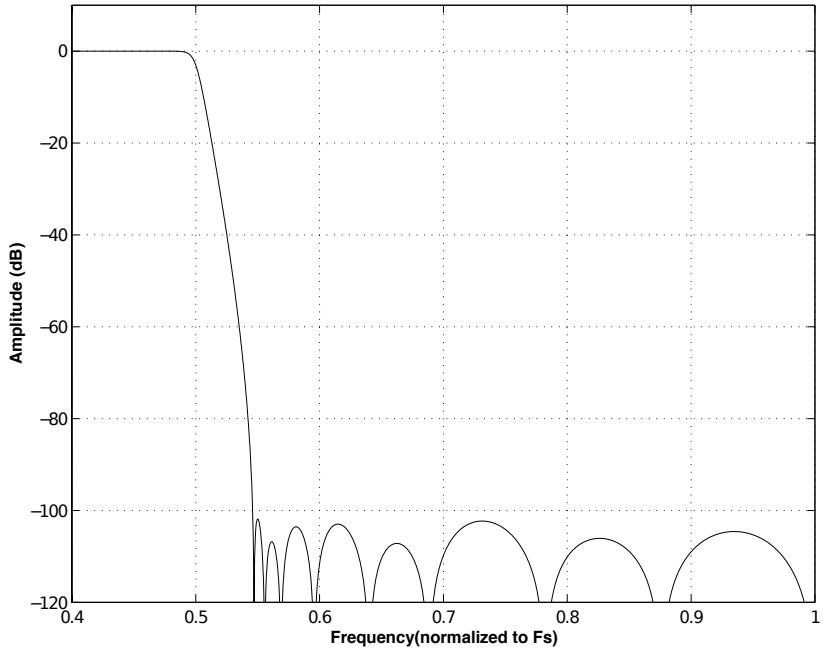

Figure 5.2: Filter response plot of the rejection band with fast roll-off interpolation filter [31]

.

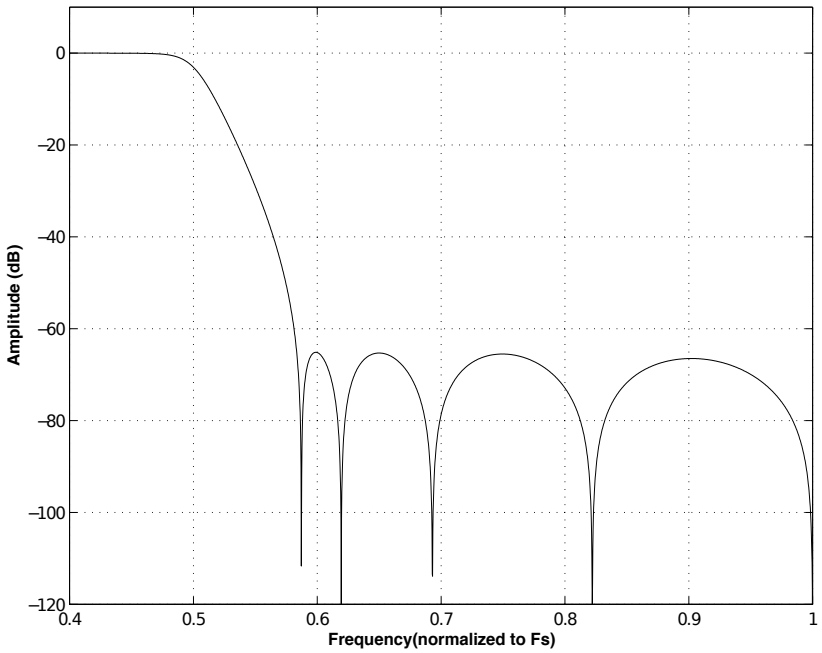

Figure 5.3: Filter response plot of the rejection band with slow roll-off interpolation filter [31]

.

5.2.3. Settings and Mode Selection

There are multiple settings that can be altered in the DAC IC. These settings are stored in the so-called *control register* and can be written using SPI (see Section 4.2.1) in software mode, or using hardware mode and connecting the right pins. In the implementation of the DAC system, the hardware mode is used since all required settings can be set. DAC pins M0 till M4 can be used to set the desired mode. The serial data format is set to Left-Justified since this will be implemented as data output from the MCU. Double-speed mode is selected because this mode supports the sample rate that will be used, see Section 5.6.6.

5.2.4. Performance

The full scale output range of the CS4384 is between $0.825\,\mathrm{V}$ and $4.175\,\mathrm{V}$. The used DACs implement a $24\,\mathrm{bit}$ resolution. They are however not ideal, and therefore the ENOB is lower than $24\,\mathrm{bit}$ (see Section 4.5). The idle channel noise is typically $-100\,\mathrm{dB}$ [31]. The maximum load noise and total harmonic distortion is $-88\,\mathrm{dB}$ with a sine of $997\,\mathrm{Hz}$ as analog output signal. Because an output amplifier is used after the DACs, the output signal of the DAC will be less powerful than the maximum output signal. In the best situation, with no load, the ENOB is:

$$\mathrm{ENOB} = \frac{100\,\mathrm{dB} - 1.72\,\mathrm{dB}}{6.02} = 16.3\,\mathrm{bit} \tag{5.1}$$

In the worst case situation with maximum load the ENOB is:

$$\mathrm{ENOB} = \frac{88\,\mathrm{dB} - 1.72\,\mathrm{dB}}{6.02} = 14.3\,\mathrm{bit} \tag{5.2}$$

This means that the ENOB will be between $14.3\,\mathrm{bit}$ and $16.3\,\mathrm{bit}$. In this calculation other noise sources such as the output amplifier are ignored.

The CS4384 has an internal reference voltage, which is used for the DAC operation. This reference voltage is proportional with the desired output voltage. If there is any error in the reference voltage, this error is translated to the output. For the application of the DAC this is not a problem since adaptive optics uses the DAC in the feedback system. If required anyway, this error can be minimized by calibrating the system by measuring the analog outputs and compensating on the digital side of the DAC for this

erroneous offset. Furthermore, the maximum capacitive load of the DAC 100 pF, so a buffer is required to achieve the system requirement of 500 pF.

5.3. Output Amplifier

A unipolar or bipolar output span is requested, but for most applications an output span of 0 V to 5 V is sufficient. An output span of ±5 V would however satisfy all cases. An amplifier with a voltage reference output is needed to reach this bipolar output span. An advantage of using this amplifier is that the maximum output capacitance of the total system becomes higher, even much greater than the required 500 pF.

As explained in Section 4.4, a configuration with an operational amplifier can be used. In our case, this is the *TL974 Rail-To-Rail Very-Low-Noise Operational Amplifier* [33]. This package contains four operational amplifiers. Therefore, 10 packages are used for all 40 analog output channels. A supply voltage of ±12 V is used to provide the amplifier power (see Section 5.4).

To implement the reference voltage in our configuration, the Quiescent Voltage output of the CS4384 DAC will be used. This Quiescent Voltage is equal to the output when no input signal is applied. Since the serial input data is in two's complement, this Quiescent Voltage indicates the middle of the output span. This voltage is approximately 2.5 V. With four resistors per analog channel it is possible to reach an arbitrary output span in the range of ±12 V. The DACs used in our configuration have an output span from 0.825 V and 4.175 V, which is 3.35 V peak-to-peak. In order to reach a 10 V peak-to-peak output span, the V_{in} voltage gain should be adjusted so that this is accomplished. With use of the transfer function of Equation (4.4) the desired output span can be expressed as:

$$V_{out,pp} = 3.35\,\text{V} \cdot \frac{R_2}{R_1 + R_2}\left(1 + \frac{R_4}{R_3}\right) = 10\,\text{V} \tag{5.3}$$

Thus,

$$\frac{R_2}{R_1 + R_2}\left(1 + \frac{R_4}{R_3}\right) = \frac{10\,\text{V}}{3.35\,\text{V}} \approx 3 \tag{5.4}$$

In this situation the offset can be calculated with the desired minimum voltage on the output of the amplifier which is 0.825 V at the input and −5 V at the output as mentioned before.

$$V_{out,min} = 0.825V \cdot 3 - 2.5 \cdot \frac{R_4}{R_3} = -5\,\text{V} \approx 2.5 - 3 \cdot 2.5 = -5 \tag{5.5}$$

Thus the ratio between R_4 and R_3 is:

$$\frac{R_4}{R_3} = 3 = \frac{3 \cdot R_3}{R_3} \tag{5.6}$$

From Equation 5.4 and 5.6 the ratio between R_2 and R_1 can be calculated.

$$\frac{R_2}{R_1 + R_2}\left(1 + \frac{R_4}{R_3}\right) = \frac{4R_2}{R_1 + R_2} \approx 3 \tag{5.7}$$

Thus,

$$\frac{R_2}{R_1 + R_2} = \frac{3}{4}. \tag{5.8}$$

Now that the ratio's between the resistors are known, the absolute values of the resistors can be chosen. Since the output impedance of the DAC influences the transfer function of the output amplifier, a sufficiently high value for the resistor R_1 should be chosen. The output impedance of the DAC is 130 Ω. On the other hand, a higher resistor value results in more noise, see Section 4.4. Therefore, a value of 15 kΩ is chosen. With this value and the ratio known, all the remaining resistor values can be calculated. The schematic of the output amplifier with all of the resistor values is shown in Figure 5.4.

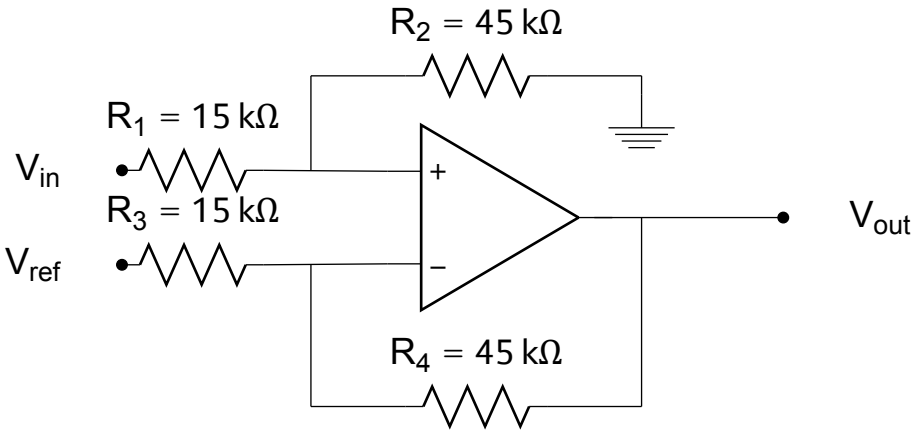

Figure 5.4: Output amplifier schematic with determined resistor values.

The spectral noise voltage level of the TL974 op-amp is $4\,\mathrm{nV}\sqrt{\mathrm{Hz}}^{-1}$ for a bandwidth from $100\,\mathrm{Hz}$ to $100\,\mathrm{kHz}$ [33]. The signal-to-noise ratio can be calculated as:

$$SNR_{amp} = 10{\cdot}\log_{10}\frac{P}{N} = 10{\cdot}\log_{10}\frac{V^2}{\text{spectral noise}^2 \cdot \text{bandwidth}} \approx 10{\cdot}\log_{10}\frac{(5\,\mathrm{V})^2}{\left(4\,\mathrm{nV}\sqrt{\mathrm{Hz}}^{-1}\right)^2 \cdot 20\,\mathrm{kHz}} \approx 139\,\mathrm{dB}, \tag{5.9}$$

where P is the signal power, N is the noise power and V is the signal voltage level. The signal to noise ratio of the operational amplifier is relatively high compared to the signal to noise ratio of the DAC. Therefore, the noise of the operational amplifier will not significantly decrease the performance of the DAC system.
The spectral thermal noise density of the $45\,\mathrm{k\Omega}$ resistor is, according to the equation for the Nyquist noise:

$$\text{spectral noise} = \sqrt{\overline{v_n^2}} = \sqrt{4k_\mathrm{B}TR} = \sqrt{4 \cdot 1.38 \cdot 10^{-23}\ \mathrm{J/K} \cdot 300\ \mathrm{K} \cdot 45\ \mathrm{k\Omega}} \approx 28\,\mathrm{nV}\sqrt{\mathrm{Hz}}^{-1}, \tag{5.10}$$

where k_b is the Boltzmann constant, T the absolute temperature of the resistor (room temperature) in Kelvin and R the resistance value in Ohm. The thermal noise of the resistors is larger than the noise of the amplifier. However, it is in the same order of magnitude. Therefore, the noise of the output amplifier will not have great influences on the noise of the total system.

5.4. Power Regulation

The FX3 Evaluation Board is powered from the USB 3.0 connection. This evaluation board includes voltage regulators that are both used for the board itself and to power the rest of the components in the DAC system. The audio DACs need specific voltages which are supplied by the USB 3.0 MCU board as well. These voltages are $5\,\mathrm{V}$, $3.3\,\mathrm{V}$ and $2.5\,\mathrm{V}$. The $5\,\mathrm{V}$ is directly supplied by USB, decoupled with a few capacitors. The $5\,\mathrm{V}$ for the CS4384 should be between $4.75\,\mathrm{V}$ and $5.25\,\mathrm{V}$ [31], this is exactly the specification of the USB 3.0 $5\,\mathrm{V}$ supply [34]. For the $3.3\,\mathrm{V}$, the voltage regulator Texas Instruments TPS74801DRCR is used, which can supply $1.5\,\mathrm{A}$ of current. The TPS76801QD voltage regulator is used to create the needed $2.5\,\mathrm{V}$ level, up to a maximum of $1\,\mathrm{A}$ of current. The voltage regulators on the FX3 board are linear voltage regulators. The efficiency equals the ratio between the output and input voltage [35]. In total, $900\,\mathrm{mA}$ from the $5\,\mathrm{V}$ USB power can be used since the FX3 board is in USB powered mode [21].

For the output amplifier a reference voltage and power supply voltage is needed. In order to reach a bipolar output span, a bipolar power supply is required in the system. For this, ±12 V bipolar supply voltages will be used. This voltage can be supplied by the DC/DC converter NTA0512MC [36] which is powered by an input voltage of 5 V supplied by USB. The efficiency is 77 % according to the data sheet. The maximum output current is ±42 mA, which is enough to power the output amplifier.

The total power supplied by USB 3.0 is up to $5\,\text{V} \cdot 900\,\text{mA} = 4500\,\text{mW}$. The total power consumption should be less than the maximum supplied value. In Table 5.1 the power consumption of the components is shown, calculated with data from the data sheets corresponding to the components. The efficiency of the DC/DC converter for the supply of the output amplifier is taken into account. The maximum output current can be calculated from the maximum output capacitance load of the DAC system, since this is the maximum capacitance that can be loaded in the time of one frame. An approximation for the maximum output current is:

$$I_{max} = C\frac{dV}{dt} \approx C\frac{\Delta V}{\Delta t} = 500\,\text{pF} \cdot \frac{5\,\text{V}}{50\,\mu\text{s}} = 50\,\mu\text{A}, \tag{5.11}$$

where C the maximum load capacitance, ΔV is maximum voltage change in one frame and Δt the duration of one frame.

The maximum total power consumption is 5211 mW. This is more than the power supplied by the USB 3.0 connection. However, this maximum power will not always be used since not all components use the maximum specified power. The DAC will not be used on the maximum sample rate of 216 kHz [31], which means that less power will be used compared to the case where the maximum sample rate is chosen.

Table 5.1: Power consumption for the components in the system, 5211 mW in total.

Supply	Amount	Voltage	Max. Current	Efficiency	Max. Power
MCU Core supply	1	1.2 V	200 mA	24%	1000 mW
FX3 USB supply	1	5 V	60 mA	100%	300 mW
DAC Analog Supply	5	5 V	92 mA	100%	2300 mW
DAC Digital Supply	5	2.5 V	24 mA	50%	600 mW
Output Amplifier pos. supply	40	12 V	0.8 mA	77%	499 mW
Output Amplifier neg. supply	40	−12 V	0.8 mA	77%	499 mW
Output Amplifier output current	40	5 V	50 μA	77%	13 mW

5.5. Signal Connections between Implementation Components

Figure 5.5 represents how the DACs are connected to the MCU.

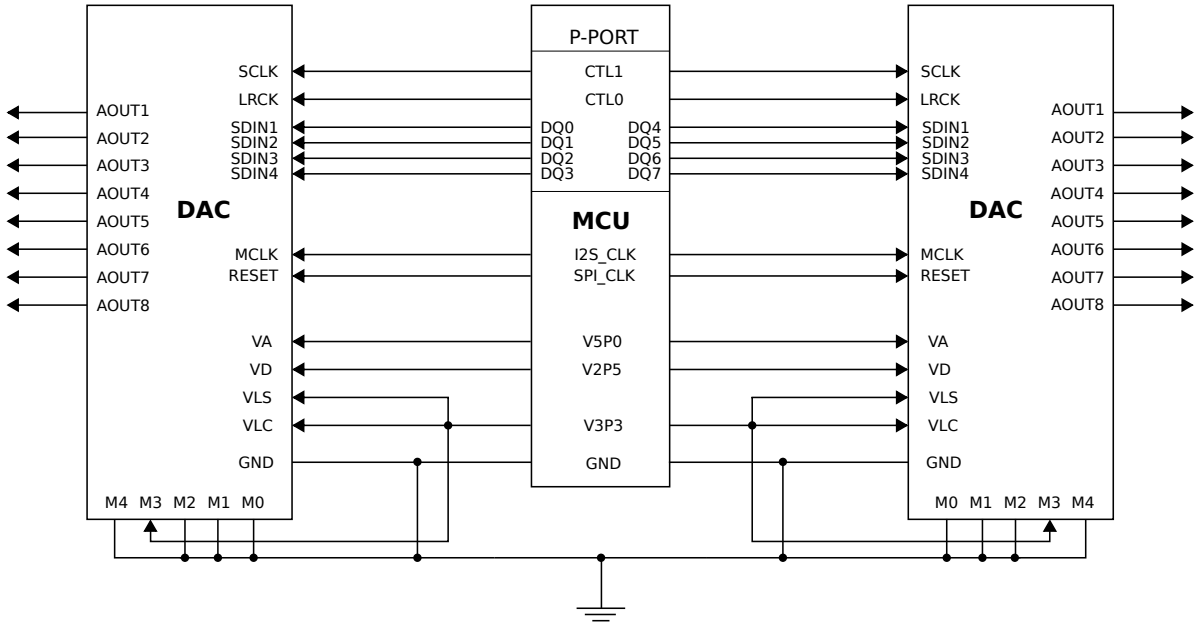

Figure 5.5: Detailed schematic of how the DACs are connected to the MCU. In this figure only two DACs are shown, however the same holds for connecting more DACs. Each line represents a signal. Port labels that are aligned to the center are signals that are shared across the multiple DACs.

For the software, two parts are required: firmware for the MCU, and drivers for the PC. The two work together to form an interface between the PC and the MCU.

5.5.1. USB Transmission Mode

Before any design choices can be made regarding the implementation of the USB connection between the MCU and the PC, the USB transmission mode needs to be determined. As described in Section 4.3.2, there are four available transmission modes: Control, Bulk, Interrupt and Isochronous. Control transmissions are not suitable for our application since they are not meant for data transfers but for protocol related transmissions only. Next, interrupt transmissions do not prove to fulfill the purpose of a quick USB connection either. This is because interrupts on USB 3.0 are (still) bound to polling, which means that there is a frame period of $125\,\mu s$ between every data transfer. This results in only a refresh rate of at most $8\,\text{kHz}$. Another alternative would be the isochronous transmission mode, but the service interval for isochronous transmissions is $125\,\mu s$ as well, again meaning only a refresh rate of up to $8\,\text{kHz}$ [20]. As a final candidate, there are bulk transfers. With USB 3.0, bulk transfers were extended with a new feature called *bulk streaming* as described in Section 4.3.3. With bulk streaming, packets of data can be subsequently sent without waiting for a next time slot. This is useful when aiming for higher refresh rates, which is the case for the DAC system. The main disadvantage of bulk transfers is the lack of guaranteed bandwidth, which might result in an unstable data connection, see 4.3.2.

Since refresh rate is an important aspect for the DAC system, bulk transfer mode will be used in the first place. A reconsideration is required when the data connection turns out too unstable.

5.6. Firmware

The MCU converts the incoming USB data into a usable data format for the DACs. The processor itself however is not capable, but the GPIF II is. Please refer to Section 5.1.1 for more information about the GPIF II. The data path from USB to the GPIF II is however not that obvious.

Multiple data paths are possible, these will be described first. Each option is then discussed using evaluations on the problems that are expected with each data path. The chosen data path is then expanded into a total implementation, which is also described into depth. The results that are found using this implementation are discussed in Section 6.1.

5.6.1. Possible Data Paths

In order to provide all five 8-channel DACs with sample data, the GPIF II is used for its ability to implement high-speed and time-critical interfaces. The exact realization of this interface in the whole system is however to be determined, options are listed below. These options are illustrated in Figure 5.6.

1. Use the internal processor to manually handle the incoming USB packets and manually send this to the GPIF II state machine. The main advantages are the flexibility and simplicity of data manipulation by using a processor and the facilities to save data. The main disadvantage is the limitation on refresh rate and increase of latency that comes with manually handling the link between USB and GPIF II.

2. Use a direct connection between USB and the GPIF II state machine, without the processor involved at all. This gains the highest refresh rate and lowest latency since the processor is removed from the data path. Although this increases the performance, the advantages of using a processor are removed. Manipulating data and saving data become major problems.

3. Use a direct connection between USB and the GPIF II state machine, but use the processor to save data. GPIF II receives the USB packets, which are then sent to both the DACs and the processor. Advantages are low latency and the facilities to save data permanently to be used as infinite buffer when the USB connection fails to supply a stable stream of data. A disadvantage is the relatively high latency between the state machine and the processor, limiting the speed of saving and retrieving data.

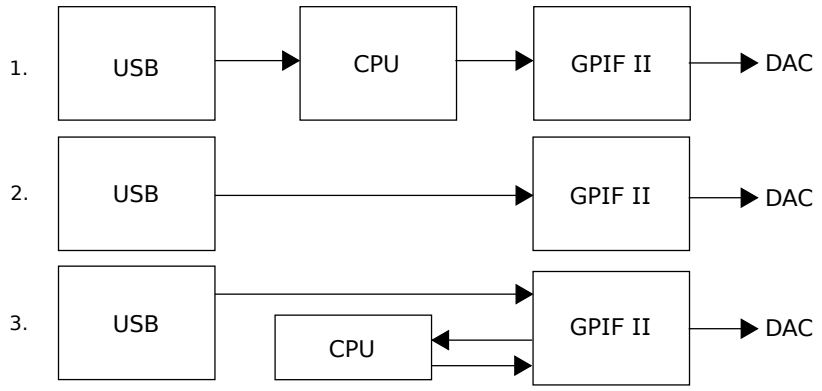

Figure 5.6: Multiple data paths are possible from USB to the DACs, all with their own advantages and disadvantages.

5.6.2. Evaluation of System Parameters for Data Path

To decide which data path to use, some parameters need to be defined. First, the ability to buffer the analog outputs for longer periods. When no USB packet is provided for a long time it is desirable to leave data buffering to the DAC itself. If that is the case, the MCU would be relieved of this task, making intensive use of the processor for data buffering unnecessary.

The CS4384's datasheet does not supply an answer to this question, so an experiment was performed. To prove or disprove buffering functionality in the DAC, the Left-Justified LRCK signal was stopped while all other signals are continued as before. As expected for audio interfaces such as Left-Justified, stopping the LRCK is not permitted and the output will be set to the default value of 2.5 V instantly.

Secondly, the USB connection reliability has to be determined. Since bulk streaming mode is used, bandwidth is not guaranteed (see Section 4.3.3). Therefore, the USB will not provide a reliable stream of data packets. The intervals between the packets will vary and gaps in the transmission will occur, making it necessary to buffer the received samples on the MCU to keep the DACs providing with data. This buffered data may be old, but is still preferred over sending the default value of 0 V to the DACs in case of a USB transmission hitch.

Lastly, the connection between GPIF II and the processor needs to be evaluated. Tests show that it can take up to 90 μs to transfer the full DMA buffer from the state machine to the processor, trigger an interrupt there and then return a buffer back to the state machine in this interrupt. This latency limits the feasibility of data path option 3 since the required 20 kHz refresh rate corresponds to a 50 μs period, which is shorter than the latency. There is thus not enough time to buffer a packet and send it back to the state machine to feed the state machine with data again.

5.6.3. Discussion of System Parameter Evaluation and Selection of Data Path

The used USB connection will not provide enough stability since bulk streaming is used. This makes data buffering necessary, but this can not be done using the DAC itself. Therefore, the MCU needs to perform this task, either with the processor or the state machine. When using the processor as data buffer, relatively large latencies between the GPIF II and the processor will cause problems in the case of option 3. Also, option 1 will be bound to these high latencies.

The direct connection between the USB and the state machine without intervention of the processor was chosen since this implementation secures the highest refresh rate with the lowest latency. Therefore, this implementation has the most potential for achieving the specifications described in Chapter 2. The problems on data manipulation and (permanent) buffering should however be solved before this can be achieved.

5.6.4. Data Interface: Left-Justified

The state machine implements a Left-Justified interface with 20 data lines: four lines per DAC. Furthermore, a master clock (MCLK), a serial clock (SCLK) and a channel selector (LRCK) are included in the state machine as seen in Figure 5.5.

The timing of the Left-Justified interface has been discussed in Section 4.2.2. The state diagram should implement the timing diagram of Figure 4.10, plus the extra MCLK which is needed for the Delta-Sigma DAC. The MCLK signal should be a multiple of the sample rate (LRCK) and be synchronous with the LRCK signal.

5.6.5. Data Format

The Left-Justified interface multiplexes the serial data lines by using the LRCK signal. This splits the 40 DAC channels into two groups of channels, each consisting of 20 channels. According to the CS4384's datasheet [31], the channels are mapped on odd and even outputs as left and right channels, respectively. One data line contains a left and right channel, which are adjacently numbered channels. Since the outputs are ordered as output pins, it will lead to a more efficient PCB design when these pins can be mapped to the output connector directly, without crossing other DAC output lines. Therefore, it is chosen to map the pins correctly in software, thus odd and even channels in separate channel groups.

The left channel is written first, when LRCK is high. Since the left channels are mapped to the odd numbered channel group, these need to be written first. This results in 20 times 24 bits of data to be sent. After that, same holds for the even numbered channels, which is sent after the odd numbered channel group.

The data bus size is required to be at least 20 bits wide, meaning 20 lines of parallel interface. This is because the data lines are used for two channels, being multiplexed by the LRCK signal. It is however not possible to implement a data bus width of exactly 20 bits: the available options are 8, 16, 24 and 32 bits of data bus width as a restriction by the MCU hardware. Since 20 is the minimum, 24 and 32

remain options. Since the remaining data bus width will not be connected, a 24 bits data bus is the best option since it makes use of the input data more efficiently. The remaining four bits of the data bus should be present in the buffer, even though these bits will be written to non-connected output pins. This results in the data format of Figure 5.7, which also clarifies the way odd and even numbered channel groups are sent.

The data format of Figure 5.7 results in a required buffer size of 144 bytes per sample for each of the 40 channels. This is calculated as follows:

$$\begin{aligned}\text{buffer size} &= \text{resolution} \cdot \text{channels per LRCK period} \cdot (\text{total channels} / 2 + \text{number of padding bits}) \\ &= 24 \cdot 2 \cdot (20 + 4) = 1152\,\text{bit} = 144\,\text{bytes}.\end{aligned} \tag{5.12}$$

USB 3.0 bulk transfers can contain up to 1024 bytes per single data packet [19], which can fit up to seven 144 bytes samples.

	20 data bits				4 bits padding			
24x24 bits with padding	Ch 1 [MSB]	Ch 3 [MSB]	⋯	Ch 39 [MSB]	0	0	0	0
	Ch 1 [MSB-1]	Ch 3 [MSB-1]	⋯	Ch 39 [MSB-1]	0	0	0	0
	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮
	Ch 1 [LSB+1]	Ch 3 [LSB+1]	⋯	Ch 39 [LSB+1]	0	0	0	0
	Ch 1 [LSB]	Ch 3 [LSB]	⋯	Ch 39 [LSB]	0	0	0	0
24x24 bits with padding	Ch 2 [MSB]	Ch 4 [MSB]	⋯	Ch 40 [MSB]	0	0	0	0
	Ch 2 [MSB-1]	Ch 4 [MSB-1]	⋯	Ch 40 [MSB-1]	0	0	0	0
	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮
	Ch 2 [LSB+1]	Ch 4 [LSB+1]	⋯	Ch 40 [LSB+1]	0	0	0	0
	Ch 2 [LSB]	Ch 4 [LSB]	⋯	Ch 40 [LSB]	0	0	0	0

Figure 5.7: Data format of a single update for all 40 channels. Should be read from left to right, top to bottom to obtain a bitstream.

5.6.6. Requirements on Serial Interface Frequencies

The CS4384 has some restrictions on MCLK, SCLK, LRCK (sample rate) and their relations. Serial input sample rates between 4 kHz and 216 kHz are accepted. These frequencies are splitted into three modes when manually selecting the mode: single-speed (4 kHz to 54 kHz), double-speed (50 kHz to 108 kHz) and quad-speed (100 kHz to 216 kHz).

The prescaler between MCLK and LRCK is defined as

$$f_{LRCK} \cdot A_{LRCK-MCLK} = f_{MCLK}, \tag{5.13}$$

where $A_{LRCK-MCLK}$ is the prescaler between LRCK and MCLK, which can be the values from Table 5.2, taken into account the sample rate (f_{LRCK}).

Table 5.2: LRCK-MCLK prescaler ($A_{LRCK-MCLK}$) options per sample rate on the CS4384 [31].

	$A_{LRCK-MCLK}$									
Sample Rate	64	96	128	192	256	384	512	768	1024	1152
4 - 38 kHz					✓	✓	✓	✓	✓	✓
38 - 54 kHz					✓	✓	✓	✓	✓	
50 - 108 kHz			✓	✓	✓	✓	✓			
100 - 216 kHz	✓	✓	✓	✓	✓					

The relation between LRCK and SCLK is

$$f_{LRCK} \cdot N = f_{SCLK}, \tag{5.14}$$

with N the amount of bits in a LRCK period (two channels), which is

$$N = 2 \cdot (N_{data} + N_{padding}), \tag{5.15}$$

with N_{data} = 24 the amount of PCM data bits per sample per channel and $N_{padding}$ the amount of padding bits per sample per channel, which is set to 24 to match N_{data}. This is done for simplicity and minimizing the amount of required states in the state machine. Only two counters are available in the GPIF II interface, so these can be efficiently used to both count the data bits and padding bits to the same constant maximum count of 24. To satisfy Equation 5.14 and 5.15, it must hold that $f_{SCLK} = 96 f_{LRCK}$. The prescaler between MCLK and SCLK now becomes more discrete:

$$\frac{f_{MCLK}}{f_{SCLK}} = \frac{A_{LRCK-MCLK}}{2 \cdot (N_{data} + N_{padding})} = \frac{A_{LRCK-MCLK}}{96}, \tag{5.16}$$

which should be an integer value. The set of possible values for $A_{LRCK-MCLK}$ now reduces to 96, 192, 384, 768 and 1152, of which the allowed values still depend on the sample rate according to Table 5.2. All speed modes allow at least one of these values, so $A_{LRCK-MCLK} = 24$ bit can be safely chosen.

As master clock, the *P-Port Interface Block clock* (PCLK) can be used on which the GPIF II works. This clock is commonly used for synchronous interfaces in GPIF II. The PCLK is prescaled from the 400 MHz internal system clock which prescaler can range from 2 to 1024, resulting in a maximum clock frequency of f_{PCLK} = 200 MHz. When using the PCLK as MCLK, the PCLK may not go above 55 MHz according to Equation 5.16. To keep as much timing freedom while designing the state machine as possible, another clock is used as MCLK: the I^2S clock from the processor. This clock is prescaled from the same system clock, making the clocks synchronous. It is set to 25 MHz and $A_{LRCK-MCLK}$ is set to 384 to obtain a sample rate of

$$f_{LRCK} = \frac{f_{MCLK}}{A_{LRCK-MCLK}} = \frac{25\,\text{MHz}}{384} = 65.1\,\text{kHz}, \tag{5.17}$$

which is in the range of double-speed mode (50 kHz to 108 kHz). The serial clock frequency becomes

$$f_{SCLK} = N \cdot f_{LRCK} = 96 f_{LRCK} = 6.25\,\text{MHz}, \tag{5.18}$$

The GPIF II is used on the maximum clock frequency of f_{PCLK} = 200 MHz. From this frequency, the SCLK has to be made using a manual prescaler. This prescaler is

$$A_{SCLK-PCLK} = \frac{f_{PCLK}}{f_{SCLK}} = \frac{200\,\text{MHz}}{6.25\,\text{MHz}} = 32. \tag{5.19}$$

Data is written on the Left-Justified data lines on the falling edge of the SCLK as illustrated in Figure 5.8. The data should be stable here, meaning that the data has to be written to before the rising edge. The falling edge is ideal since the adjacent rising edges are on equal distance.

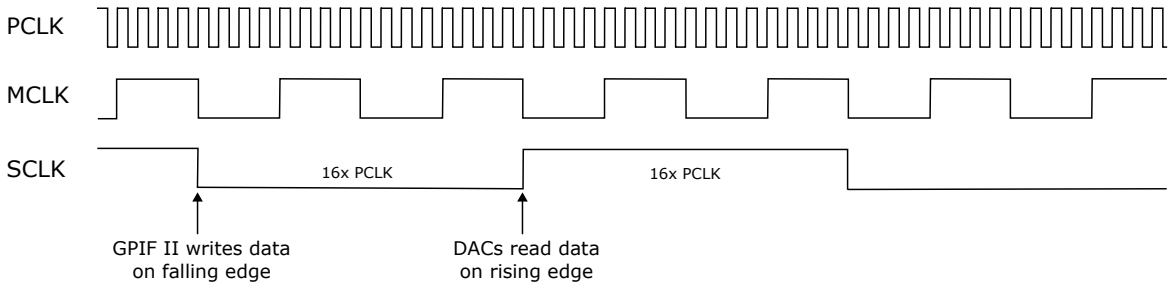

Figure 5.8: The highest clock frequencies in the system. The GPIF II clock frequency f_{PCLK} = 200 MHz, the DAC master frequency f_{MCLK} = 25 MHz and the DAC serial clock frequency f_{SCLK} = 6.25 MHz.

5.6.7. Direct Memory Access

The input data for the state machine is provided by *Direct Memory Access* (DMA). DMA works in parallel with the processor to enable read and write operations with minimal interference from the processor. Any DMA channel has a producer and a consumer. A producer writes data to the DMA channel and the consumer reads this data from the DMA channel. A DMA channel is always unidirectional, so two DMA channels are required to implement bidirectional data transfers.

All data transport on the written firmware is done by DMA. Please refer to Figure 5.9 for an overview of the DMA channels, their producer-consumer pairs and the context of the DMA channels. The USB-GPIF II pair provides the data transport of new samples. The GPIF II-GPIF II pair is used for buffering samples, which is further explained in Section 5.6.8.

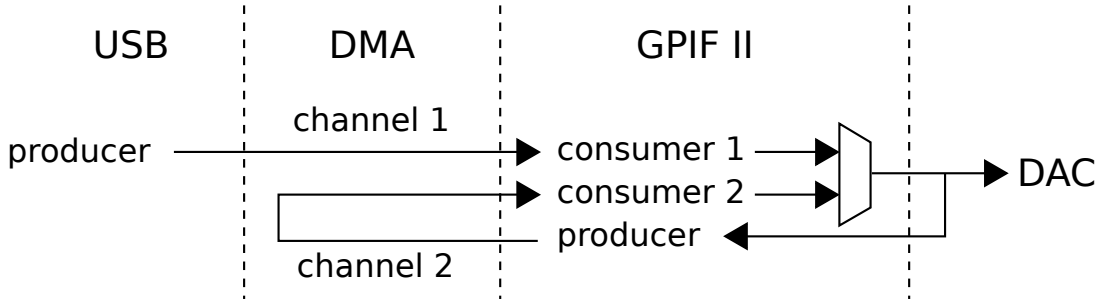

Figure 5.9: DMA consumer and producer links in the system. DMA channel 1 links the USB and GPIF II, providing new samples. DMA channel 2 loops back the GPIF II to facilitate buffering.

Both channels use two buffers, Figure 5.10 illustrates this. One producer and one consumer use two shared buffers to read and write from. Each buffer is first filled by the producer until it is full or the producer manually commits the buffer. Then, the consumer can read the buffer, preventing the producer from writing to that buffer until the consumer emptied the buffer. The producer will be stalled until the buffer is empty. The buffer is therefore mutually exclusive.

To prevent stalling of the producer, a second buffer is added. This way, the producer can write to the other buffer when the consumer reads from the first. When the consumer always empties the first buffer before the producer is done filling the second, no stalls on the producer are ever introduced. In the case of the USB-GPIF II producer-consumer pair, this reduces the USB stalls, increasing the refresh rate.

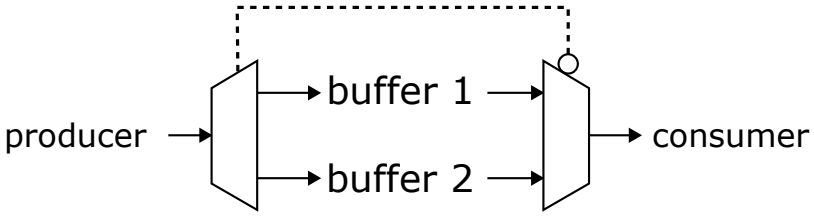

Figure 5.10: DMA channel with two buffer to reduce stalls on producer side. Buffers are mutually exclusive to producer and consumer.

5.6.8. Data Buffering

The state machine has the limitation that it can not buffer any data on its own. All data that is read from the DMA buffer is removed. When an interval between two USB packets is too big, the state machine has no data to write anymore. USB bulk transfers do not guarantee a steady stream of data packets, so buffering is required. Please refer to Figure 6.2 and Table 6.1 for statistics on the stability of the used bulk stream.

Since buffering in GPIF II is not evident, an extra DMA channel is introduced. Figure 5.9 illustrates the context of this DMA channel, linking a GPIF II producer to a GPIF II consumer. The producer reads back data from the GPIF II data bus on the rising edge of the SCLK, where the DACs read data as well, see Figure 5.8. Sample data that has just been written to the DACs will still exist on the data bus, which is then read back by the producer. These 24 read bits are added to the buffer and when the buffer is full, it is transported to the consumer, which is the state machine as well. That way, a loop is created that facilitates buffering.

When new USB data arrives, this data should be written to the output instead of the buffered data. This DMA channel is therefore implemented with a higher priority. This new sample is then buffered instead of the old one, see Figure 5.11.

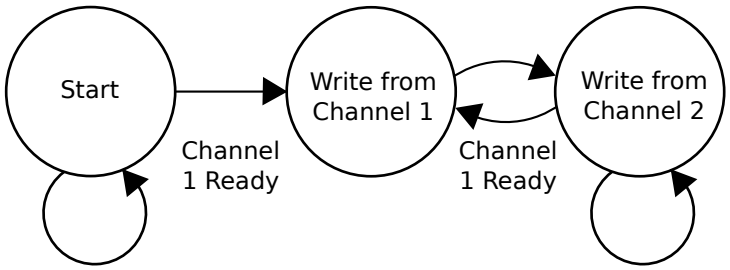

Figure 5.11: Data buffering is performed by looping back data using DMA. When new data arrives, it gains priority. State transitions can only occur on the rising edge of LRCK. Arrows without text indicate unconditional state transition.

5.6.9. Buffer Truncation

The solution found in Section 5.6.8 brings up another problem: buffers can not be truncated easily in the state machine. When new data arrives from USB to the state machine, old data is still buffered in the DMA channel linking the GPIF II producer and a GPIF II consumer. The new data should be written as soon as possible to limit the latency, but this prevents the other buffer from being emptied. In that situation, the old buffer will be written to the output after the new sample is written, which will then be sampled and buffered again by the DMA producer, causing the old samples to be re-transmitted repeatedly.

To solve this problem, the 24 padding bits are used. As described in Section 5.6.6, Left-Justified allows padding bits after the data bits. These bits are not processed by the DACs and can therefore be used to write old samples. Timing issues come up when the truncation happens too slow. Therefore, this is not performed on the SCLK frequency as happens for writing normal data, but as many PCLKs in a SCLK

are used to speed up the truncation. Due to overhead in the state machine (not being able to write every PCLK), it was only possible to implement up to 24 write actions per SCLK period of each data bus width of 24 bits. A full buffer contains 144 bytes (see Section 5.6.5), which can now be truncated in $144/24/3 = 2$ SCLK periods. Therefore, only one padding of 24 bits is required as illustrated in Figure 5.12.

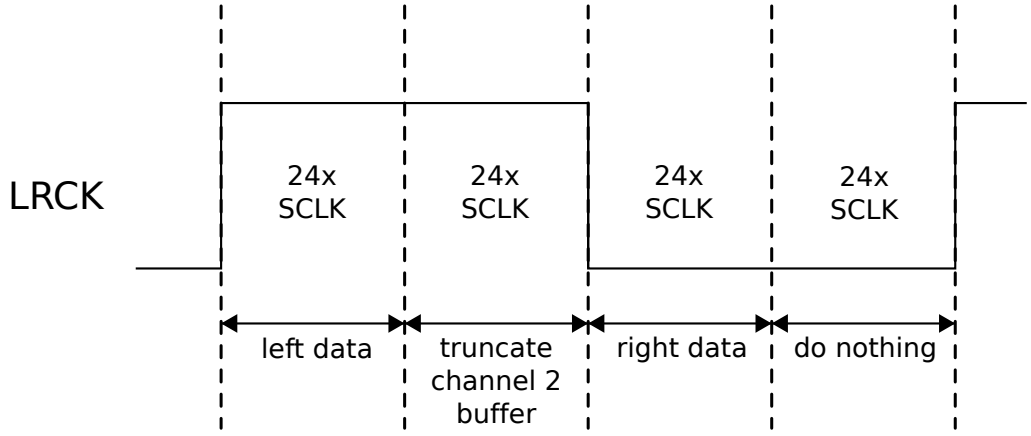

Figure 5.12: Old samples from the buffer are truncated in the padding of the first LRCK channel, where the LRCK is high.

5.7. Driver (PC)

The driver sends USB messages to the firmware which includes either data or settings. Data needs to be sent in the format as described in Figure 5.7. A digital representation of the required analog values will be sent from the PC driver. In the software the limits for the output span can be defined with a maximum of the total output span of $\pm 5\,\mathrm{V}$.

Cypress made an *Application Programming Interface* (API) for Windows, Mac OS X and Linux. For Windows, both C# and C++ variants are available. This API is a high-level interface to the CyUsb3.sys device driver which comes with the installation of the Cypress programming environment. C++ was chosen over C# since it is recommended by Cypress for higher throughput [37].

The Cypress API includes functions to find USB devices by Vendor ID and Product ID. The USB endpoints from these devices can then be used to read from or write to using the transfer type that fits the endpoint. The endpoint is the connection to the USB pipe, which enables communication with the USB device. Writing to endpoints can be done in two ways: synchronous and asynchronous. Synchronous is the easiest way to write to an endpoint, but does not enable queuing messages. Asynchronous is more difficult, but the possibility to queue messages increases the throughput. The disadvantage of queuing is that the data is not real-time anymore: the latency of a single message in the queue increases. Since the DAC is used in a feedback loop, latency should be kept as low as possible. Therefore, the synchronous method is used.

6

Results and Discussion

6.1. Results

The measurements of the complete DAC system prototype are reported in this chapter. The measurements are done to several subsystems of the prototype. For USB, measurements will be reported that describe connection performance between the MCU and the PC. Any MCU related results are reported as well, including output waveforms of the serial data transmissions and the ways the MCU can communicate with the subsystems. Next, any measurements on the DAC ICs are shown.

Figure 6.1 shows the measurement setup to analyze the output waveforms of the MCU and the analog output of two DAC channels. These measurements are performed by a logic analyzer that is capable of analyzing both digital and analog signals. The two measured output channels are the left and right channel of the first stereo audio channel, which are both supplied by the first serial data line. Any potential delay between the output of these two channels can therefore be measured.

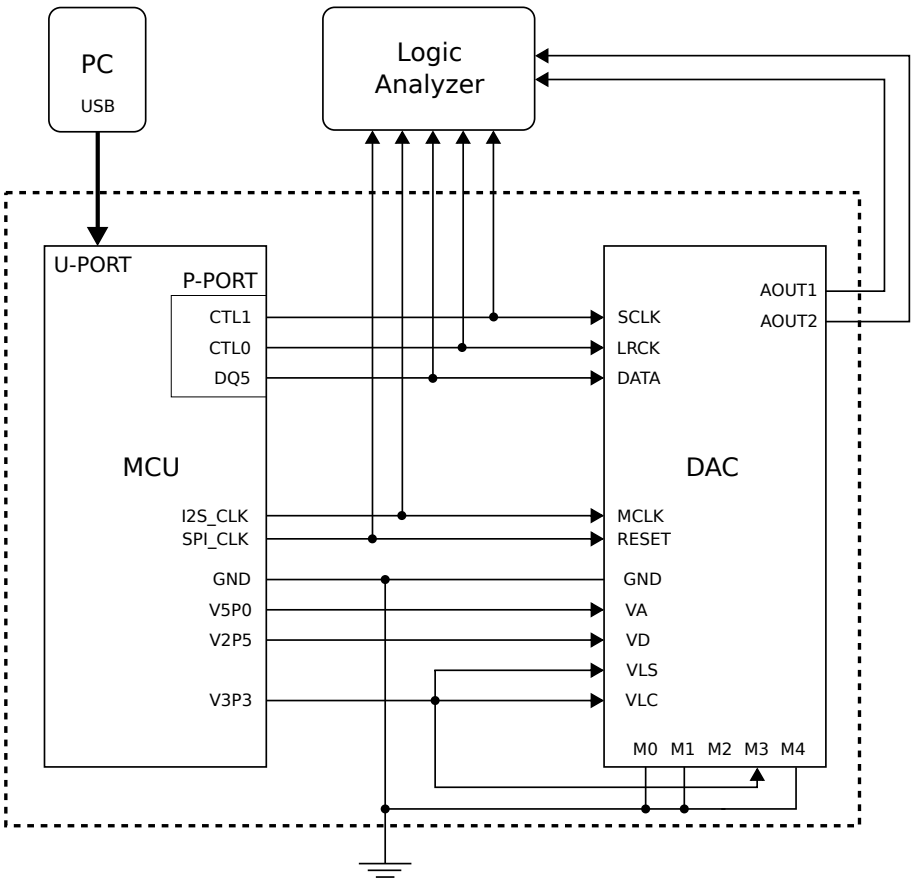

Figure 6.1: Measurement Setup for the prototype. The MCU is connected through a USB connection with the PC. The logic analyzer is capable of measuring both digital and analog signals. The ground of the system represents the ground of the USB connection.

6.1.1. Connection Performance of Universal Serial Bus

A USB 3.0 connection between a host PC and the MCU has been set up. This section reports the measurements related to the performance of the USB connection. This includes tests of the achieved sample rate and results from applying queueing in the system.

Results on Chosen USB Connection

The USB connection average refresh rate and reliability has to be determined. USB bulk streaming is used, which does not have a guaranteed bandwidth or predefined refresh rate (see Section 4.3.3). Instead, data packets are sent as quick as possible. This results in varying intervals between the packets at the USB device. These tests are performed from the PC to the GPIF II state machine, where the packet arrival interval is analyzed on both the mean and the deviation. Every packet arrival triggers a state machine output to toggle, which interval can then be measured using a logic analyzer.

The experiment was performed using 50 million USB packets, which takes about 42 minutes at the resulting 19.8 kHz refresh rate. Please refer to Figure 6.2 for the full results of this USB performance test. Additionally, outlier information is given in Table 6.1.

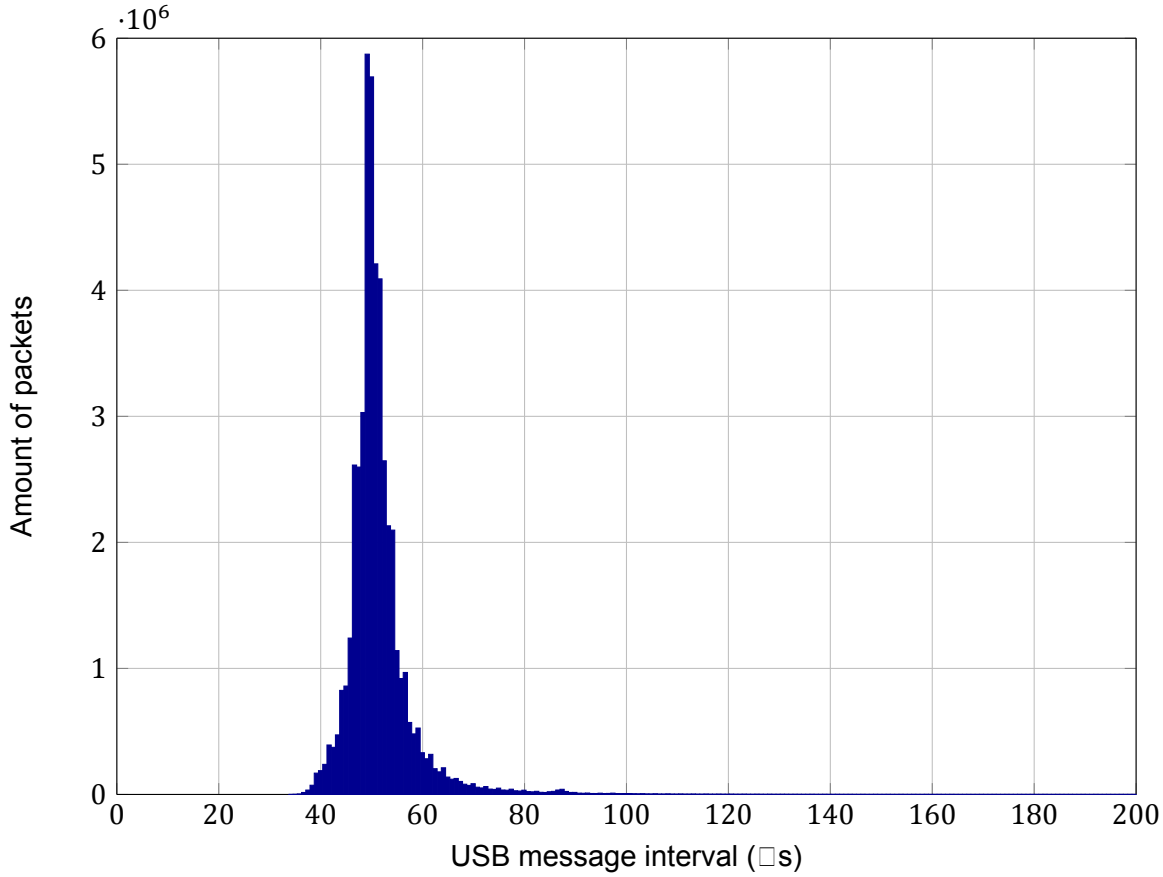

Figure 6.2: USB 3.0 received message interval in the GPIF II state machine, which is used to determine the stabilty of the bulk stream. An average of 19.8 kHz was achieved over 42 minutes. See Table 6.1 for outlier data.

Table 6.1: Outliers to USB 3.0 packet interval, supplements Figure 6.2.

Range	Occurs once per ... samples on average
> 100 □s	199
> 200 □s	641
> 400 □s	1.778
> 600 □s	3.781
> 800 □s	7.541
> 1.000 □s	13.236
> 5.000 □s	1.450.383
> 10.000 □s	11.965.656

Results from Application of Queuing

In order to gain refresh rate, USB packet queuing can be used in bulk streaming mode. The test setup used to measure the used USB connection was used, but this time queuing was used. See Figure 6.3, supplemented with outlier statistics from Table 6.2. These results were acquired with the test application from Cypress with the same firmware as in the earlier test. This application implements efficient bulk streaming, which minimizes the protocol overhead. Larger queues result in higher refresh rates but increases the latency (as packets await their turn in the queue). The maximum queue size of 64 was chosen to prove that the maximum refresh rate is 40 kHz on average, or a 25 μs message

interval. Compared to the test case without queuing, the refresh rate doubles, but the interval outliers become worse. This could be explained by the fact that the streaming protocol is optimized on solid data streams, interruptions of this stream therefore result in longer recovery times.

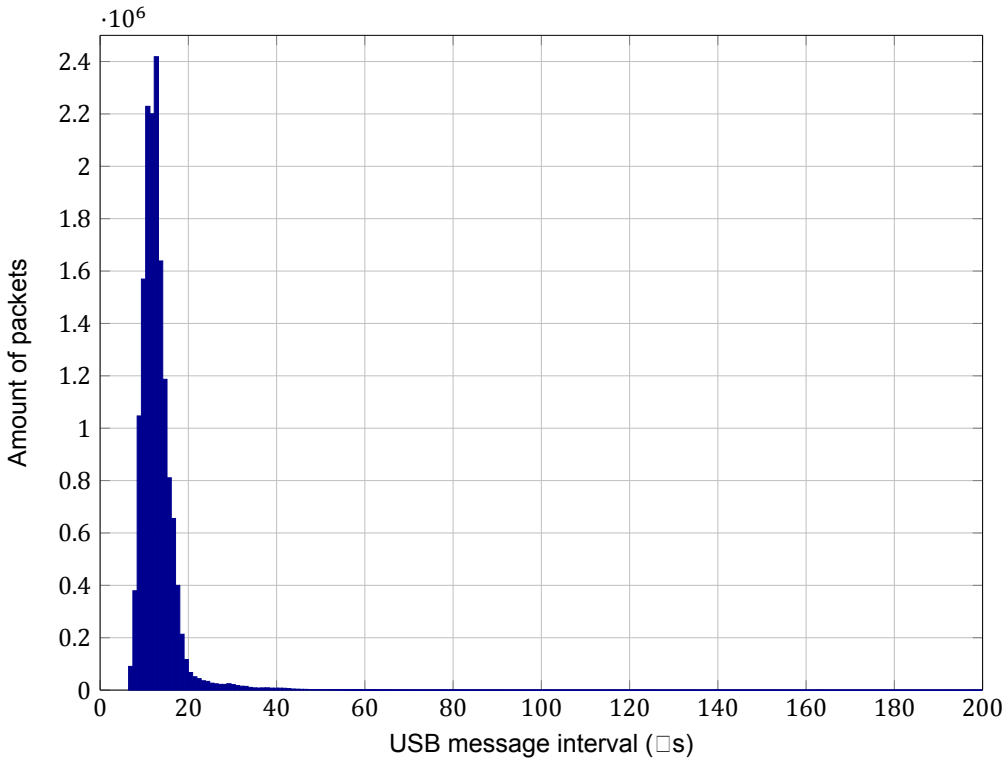

Figure 6.3: USB 3.0 received message interval when using a 64 packet queue. See Table 6.2 for outlier data.

Table 6.2: Outliers to USB 3.0 packet interval when using a 64 packet queue, supplements Figure 6.2.

Range	Occurs once per ... samples on average
> 100 μs	62
> 200 μs	63
> 400 μs	64
> 600 μs	95
> 800 μs	285
> 1000 μs	461
> 5000 μs	2.490.384
> 10 000 μs	9.961.535

6.1.2. Microcontroller (MCU)

To test the GPIF II state machine, the serial output of the MCU is analyzed. This is done by a logic analyzer connected to the Master Clock, Serial Clock, Left/Right Clock and Data lines. The output of this analysis is depicted in Figure 6.4, where one period of the left/right channel clock is shown. This means that the DATA line is valid for the left channel when LRCK is high, and valid for the right channel when LRCK is low.

Figure 6.5 shows the performance of the GPIF II state machine, which is related to Figure 6.2 which

illustrates the input stability for the state machine. The input performance of USB is made discrete since the sample rate to the DAC is constant and refreshes only happen on new sample periods.

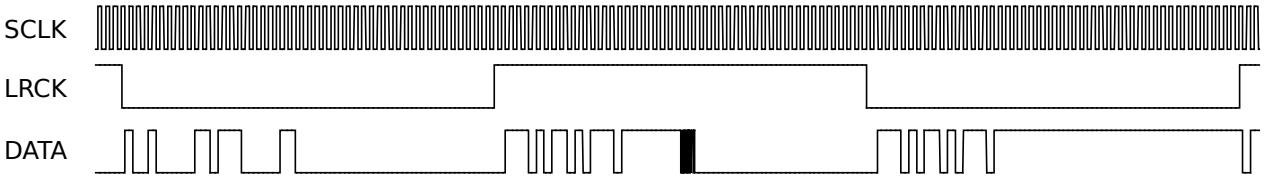

Figure 6.4: Logic analyzer output showing the Left-Justified output from the state machine. A buffer truncation is shown as well: old samples are truncated at high frequency in the padding space of the Left-Jusfified data line.

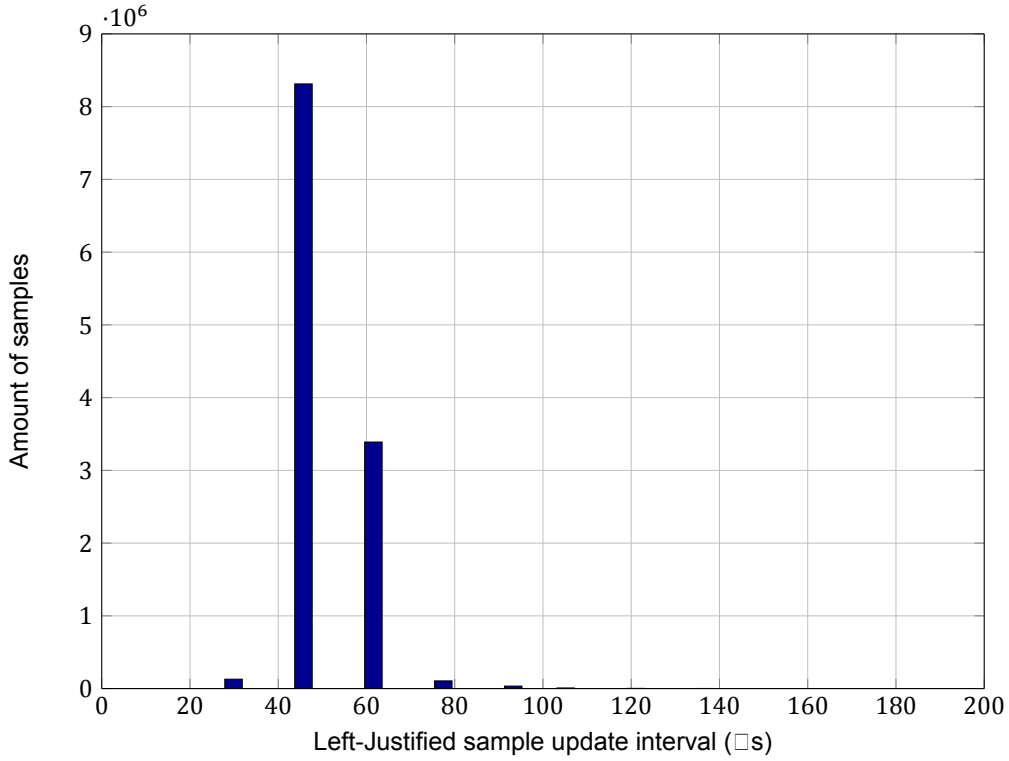

Figure 6.5: Left-Justified sample refresh interval from the GPIF II, this measures the stability of the data stream to the DACs. An average refresh rate of 19.8 kHz was achieved on average over 10 minutes.

6.1.3. Digital-to-Analog Converter

The DAC is tested in the configuration explained in Chapter 5. The serial data generated by the MCU is used as input for the CS4384. This resulted in a working DAC system. With the MCU, all the 5 DACs with in total 40 channels could be controlled, however since all the 40 channels work in the same manner this section will focus on only one channel. If a value in two's complement in Left-Justified format is fed as input on the CS4384, the analog output changes. The digital code is directly converted into an analog value since the CS4384 does not store the digital value in a register. Therefore a stream of digital data in Left-Justified format is continuously sent from the MCU. For all 40 channels these values are converted into analog values. One analog output channel is shown in Figure 6.6, which was obtained when the digital input is changed from a high (4 V) to low (1 V) value. The moment the input of the DAC changed is marked with a dashed vertical line. This change in the output is made in approximately 90 μs, but oscillations occur before and after the change. These oscillations occur since high frequency signals, such as block waves, are suppressed by the fast roll-off interpolation filter built inside the audio DAC.

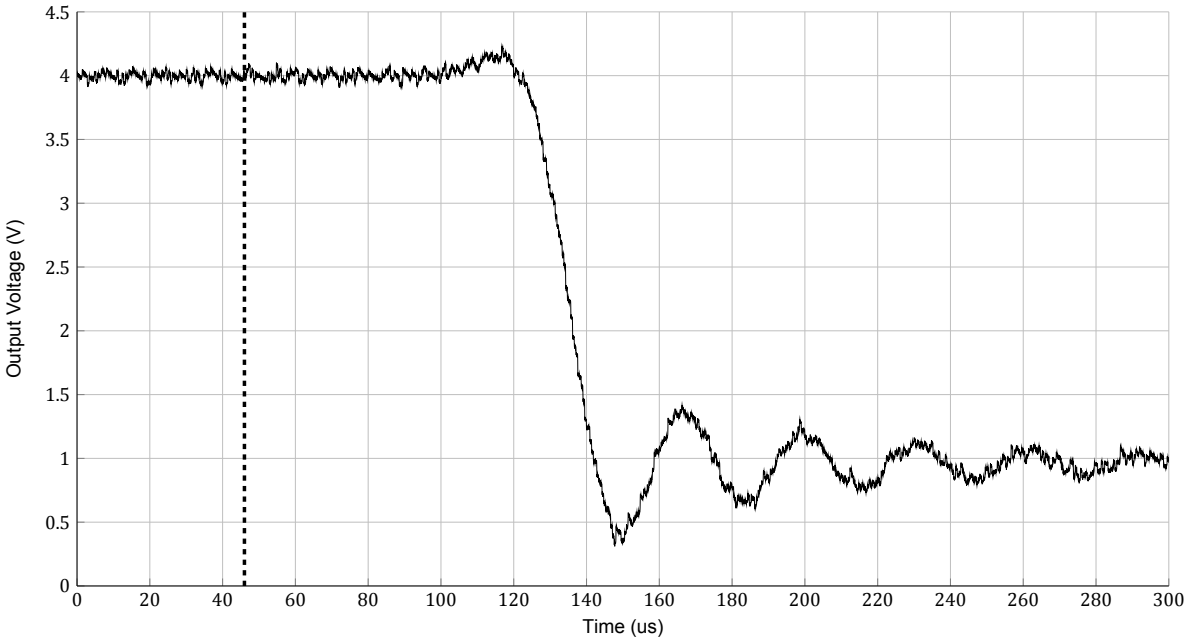

Figure 6.6: Output of the DAC when the input is instantaneously changed from a high to low value. The vertical dashed line represents the moment where the GPIF sends the serial data to the DAC.

It was also tested whether the output channels are synchronously updated. Both were sent the same square wave signal, see Figure 6.7. The figure shows that both channels are updated simultaneously.

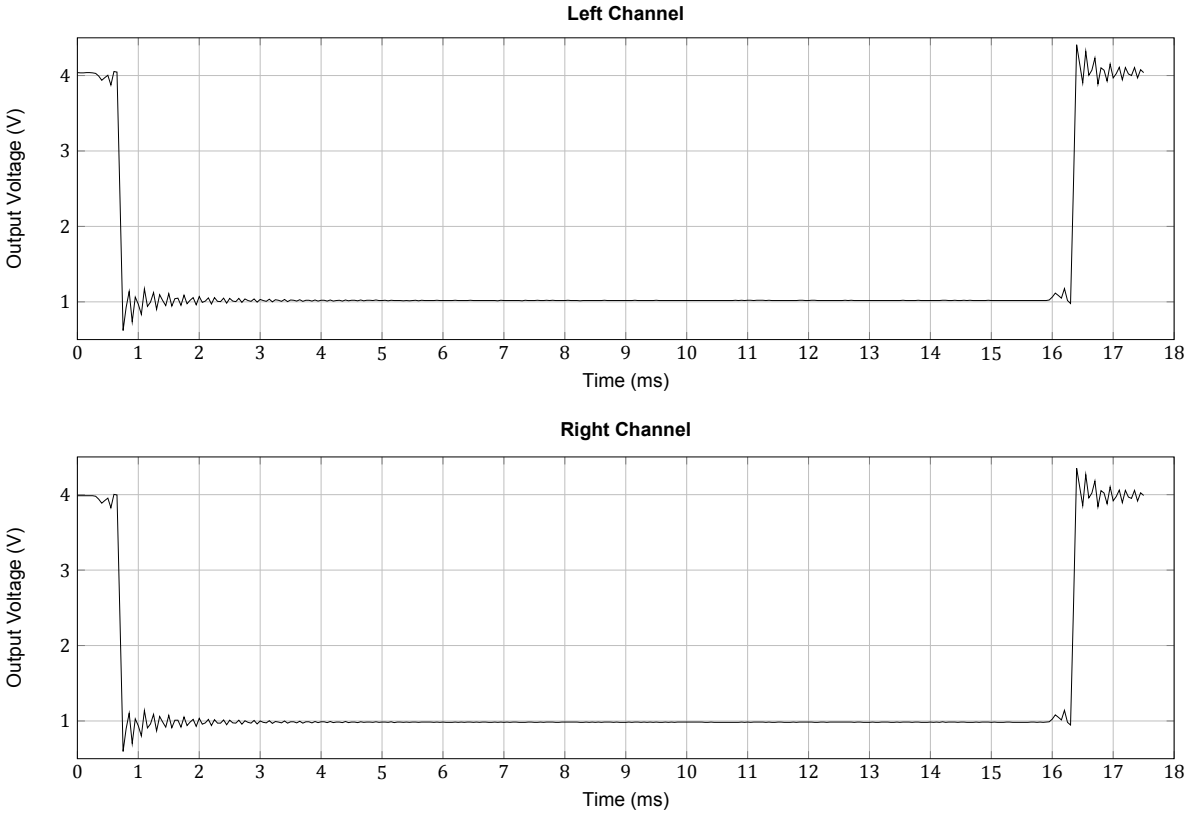

Figure 6.7: Voltage output of a DAC channel pair. The values for both channels is sent sequentially and the output is updated synchronously.

The resolution of the DAC is not measured since precise measurement equipment would be needed. Furthermore, it can be assumed that the resolution specified in the data sheet is the actual DAC resolution. Therefore, the resolution is 24 bit with an ENOB between 14.3 bit and 16.3 bit.

6.1.4. Estimation of Average Total System Latency

The latencies of the subsystems USB, GPIF II and DAC are summed into a total system latency. For USB, no direct measurements can be performed on latency. Instead, the latency is estimated on the interval of USB packets. The next packet is sent when the current packet arrival is confirmed on the PC. Therefore, this is the maximum average latency as the confirmation message back to the PC is also included in this latency. This latency of 50 μs will be used since it is hard to correct the value for the confirmation message latency.

The GPIF II outputs Left-Justified samples to the DACs on 65 kHz. Incoming samples from USB can not be processed until the start of a new sample period. This is the maximum latency that occurs in the GPIF II. The average latency is estimated to be around half of this value since USB packets arrive halfway the current sample on average. Therefore, the average latency of GPIF II is estimated to be 8 μs.

The DAC uses Delta-Sigma modulation, which always needs to be low-pass filtered. This low-pass filter prevents the output from changing fast, which poses a latency. Besides that, 55 μs is required before any change is made to be output at all, see Figure 6.6. This latency is constant for all inputs and is therefore used for the latency estimation. For small changes per sample, the introduced oscillation will be negligible. In that situation, the latency of 55 μs is correct. This will not be the case for larger steps as in Figure 6.6.

Summing these values results in an average latency of 50 μs + 8 μs + 55 μs = 113 μs.

6.2. Discussion

The refresh rate performance is tested under ideal conditions. This means that the buffer was created before the send cycle actually started. Furthermore, the PC was not used for any other purpose other than testing the system. When calculations need to be performed and the buffer is created in real-time, then the performance would drop. Also, the PC partly determines the refresh rate. When the PC was switched to power saver mode (which drops the CPU frequency to 0.7 GHz), the sample rate decreased to 8 kHz. However, this also means that a higher refresh rate might be achieved when higher performance computers are used to run the driver on.

Furthermore, a 24 bit DAC system is built, although the ENOB is only between 14.3 bit and 16.3 bit. It is debatable whether a high resolution is useful when the ENOB is lower, since when the digital input is changed with a few LSBs this will not be significant enough to see on the output. On the other hand, if a DAC is used with a lower resolution then in general an even lower ENOB results.

7

Ethics

One would not immediately think about any consequences regarding the development of a DAC. However, the applications of a High-Speed DAC like this one could sprout an ethical debate. First, the consequences of the usage of a High-Speed DAC are discussed and the solution to any ethical dilemma is proposed. Afterwards, the ethical aspects regarding anything that has to do with the production process of the DAC is discussed. As a conclusion, any environmental considerations are discussed.

7.1. Applications

A DAC has many different applications. These applications will now be individually discussed.

7.1.1. General

An ethical analysis between costs and quality, including safety, could be made for implementing DACs on a commercial scale, that is when making profit is one of the main reasons of investing into DACs. This means that there should be made a decision between the quality of the DAC and the low costs of the DAC. Let's say that a cheap DAC is used that is not always that precise. In a situation where a high powered laser is connected to the DAC, this could be a potential hazard for the operator of the system. Especially when the laser hits another target by accident because the measurement was not set up correctly. When it is the case that things like this could happen, it might be a wiser choice to invest in a better quality, and more expensive, DAC.

7.1.2. Audio

Audio is probably the most innocent application of a DAC. It should not have any bad consequences other potential than hearing loss. However, making affordable DACs for audio does bring a convenience to the general public. There is not really a downside to this, and thus does a DAC contribute to the greater good of the population which is a good thing seen from a utilitarian perspective.

7.1.3. Telescopy

An example of an application of a DAC on a bigger scale is in telescopy. Take for example the Keck Observatory in Hawaii which is pictured on the cover of this thesis. The two telescopes at this observatory make use of adaptive optics in their telescopes so that any image is as clear as possible.

This brings us to the ethical discussion of observing the environment around Earth: space. Are we that curious to explore space, and how many people would profit from any discoveries? It is clear that there is a big cash flow going to extraterrestrial research. One could say that the money going to this research could have better destinations to go to. According to consequentialism, more specifically looking from a utilitarian point of view, it would not be that viable to invest a fortune in space research since this investment does not contribute to the greater good for the majority of the population.

It is also possible to consider the use of telescopes in the reverse way: from space to Earth. Nowadays almost the whole world is photographed and can be viewed publicly on the internet. These telescopes

are carried by satellites orbiting around Earth. One could ask the question: if the technique of adaptive optics improve, what consequence could that have for the quality of the images that the telescopes are able to make, and when could we see this as a violation of privacy when someone is able to take a look into a random person's backyard?

7.1.4. Measurement Industry

The usage of DACs in the measurement industry has the possibility to lead to more accurate measurement results that would improve the research in a specific subject. Of course, Research & Development is a very wide sector with many fields of research. One cannot easily tell for what purposes the DAC will be exactly used, but one can keep in mind that the purpose is not always ethical or for the greater good.

7.2. Production Process & Environment

After the DAC is designed, the system will be produced on a large scale. There are a couple of ethical issues involved in this production process. A consideration between production cost, safety for the workers and any environmental impact should be made. For example the product should be *Restriction of Hazardous Substances* (ROHS) compliant. This means that some substances that are dangerous for the environment should not be used. From deontologic point of view, even though the environmental impact is unknown, the risk for severe problems should be avoided at all times. Furthermore, the factory where the products are manufactured should respect their workers and give them a safe working environment.

8

Conclusion and Recommendations

8.1. Conclusion

The following conclusions can be made from this project.

A working prototype of the proposed DAC system is implemented and tested. The prototype is USB powered. However, the maximum power that all the components consume together is higher than the total power supplied by the USB 3.0 connection. Therefore it is not guaranteed that the USB can supply enough power for the system with the chosen components. A stable connection between the USB bus of the PC and the DACs is accomplished. This connection is stable enough to prevent any gaps in the data stream in the case of a USB miss or malfunction. This results in a stable input for the DACs and thus a stable analog output signal.

Furthermore, a refresh rate of 19.8 kHz is achieved where all analog output channels are updated simultaneously. From measurement results we can conclude that the average latency between a digital signal from the PC to an analog output of the DAC system is about 113 μs. This however highly depends on the input signal. This latency covers the latency of the total DAC system.

The analog outputs of the DACs do not show a perfect, undisturbed waveform. The main reason, besides any noise, is that the implemented DAC ICs feature filters that are specifically meant for audio applications. These filters cause an oscillating behaviour in the analog outputs. To minimize this unwanted behaviour, the sample rate of the serial interface is increased. The final sample rate for the serial interface is chosen to be 65.1 kHz.

The DAC system has a resolution of 24 bits. This meets the requirement of a 16 bit resolution. Theoretically, the ENOB is between 14.3 bit and 16.3 bit.

For the maximum load capacitance of the DAC system, a minimum of 500 pF is required. This requirement is satisfied with the output amplifier. The output span is ±5 V, which satisfies the initial requirements.

8.2. Recommendations

The synthesis of a High-Speed USB 3.0 DAC is a complex process that features multiple fields of engineering. A couple of recommendations of aspects that could be improved are described here. The product described in this thesis can merely be seen as a proof-of-concept that plays a role in a bigger goal to globally implement multi-channel digital-to-analog converters with higher refresh rates and resolution.

An idea for future work is to design an actual PCB and thus to step off from an evaluation board. This induces a more permanent solution and is less suitable for prototyping. However a PCB could highly reduce noise in the signals, thus improving the accuracy of the DAC system significantly due to a lower amount of signal loss between system components.

Furthermore, possibly a DAC IC could be used that has a better noise performance. Being too far in the design process described in this thesis, there was no time left to replace the DAC IC by an IC that performs better. A DAC that is quite the same as the Cirrus Logic CS4384 but with a better noise performance is the PCM1690 by Texas Instruments [38]. It features the same way of interfacing as the Cirrus DAC, but the noise performance is better. The CS4384 has a total harmonic distortion + noise value of typically $-88\,\mathrm{dB}$ while this value for the PCM1690 is typically $-94\,\mathrm{dB}$, which is a difference of $-6\,\mathrm{dB}$.

Since the maximum USB bulk transfer packet size is not fully utilized in the proposed DAC system, more efficient use of this space could be made by sending another sample frame in the same packet. This of course does not improve the real-time latency, but could prove useful for certain applications.

When the optimal goal is to improve the refresh rate then the option of queuing can be considered. As shown in Figure 6.3, refresh rate can be significantly increased when queuing packets in a stream transmission.

In order to improve the accuracy of the DAC even more, a feedback loop between the output of the DAC and the MCU could be constructed. By sensing the output of the DAC in combination with a known input value, an offset can be calculated. A calibration of the DAC system via software or possibly hardware could be made when this offset is known.

The maximum power that all the chosen components consume together is higher than the maximum power supplied by the USB 3.0 connection. This problem can be solved by using the more efficient *buck step-down regulators* instead of linear voltage regulators. The efficiency can be easily improved to 90% [35]. With this improved efficiency the power supplied from the USB 3.0 connection will always be sufficient.

A

GPIF II DMA State Machine

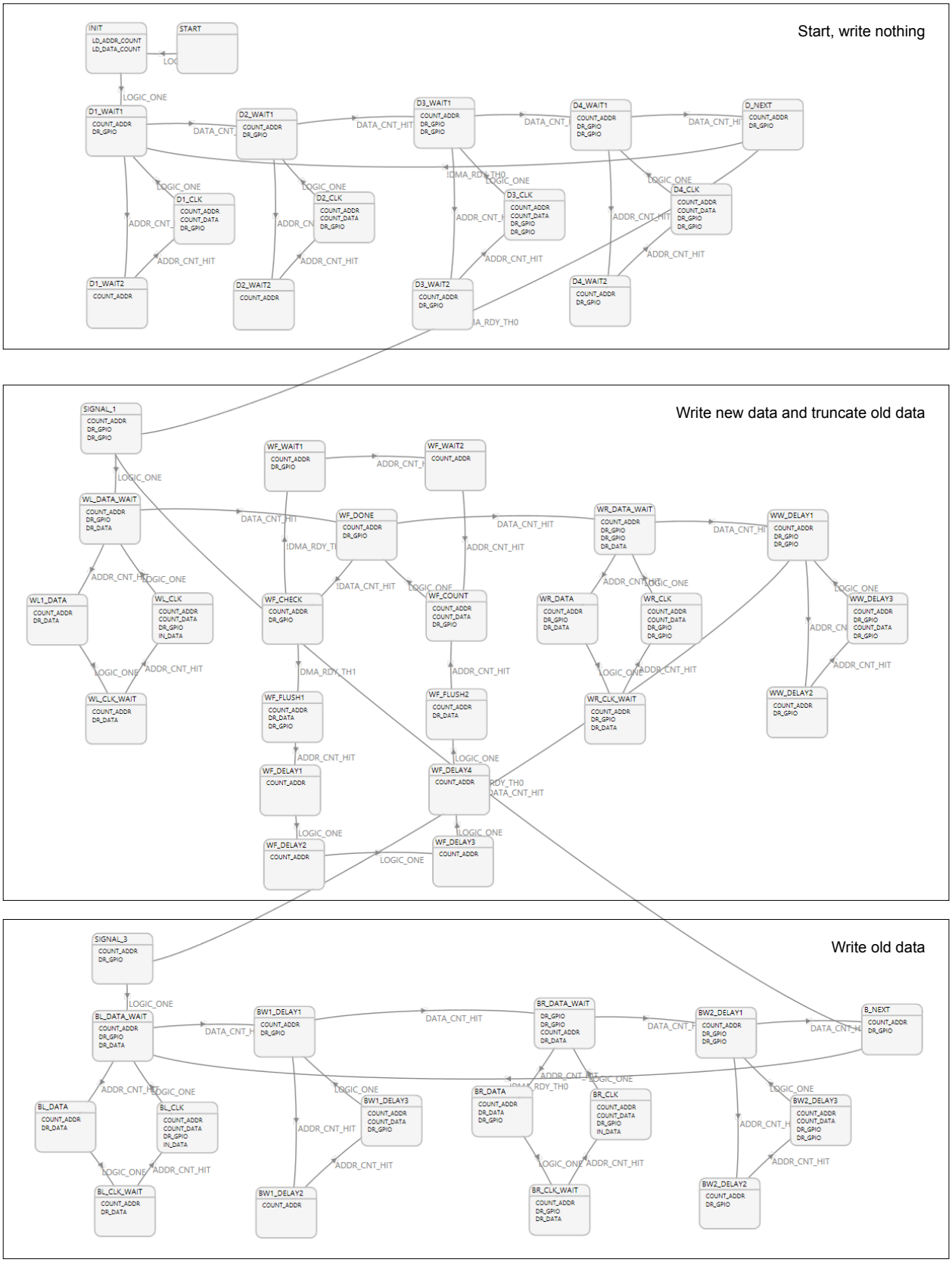

Figure A.1: The total GPIF II state machine, designed in GPIF II Designer by Cypress.

Bibliography

[1] P. O'Reilly and C. El-Khoury. (2012, March). Modern DACs and DAC Buffers Improve System Performance Simplify Design [Online]. Analog Dialogue. Available: http://www.analog.com/library/analogDialogue/archives/46-03/dacs_buffers.pdf.

[2] Walter Kesler. (2009). Basic DAC Architectures II: Binary DACs [online]. Available: http://www.analog.com/media/en/training-seminars/tutorials/MT-015.pdf.

[3] Walter Kesler. (2009). Basic DAC Architectures I: String DACs and Thermometer (Fully Decoded) DACs [online]. Available: http://www.analog.com/media/en/training-seminars/tutorials/MT-014.pdf.

[4] J.M. Verhoeven, A. van Staveren, G.L.E. Monna, M.H.L. Kouwenhoven and E.Yildiz. (2004). Structured Electronic Design. ISBN: 1-4020-7590-1.

[5] S. Tilden and Solomo Max. (2011). IEEE Standard for Terminology and Test Methods of Digital-to-Analog Converter Devices [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6152113

[6] A. Oppenheim and R. Schafer. (1975). Digital Signal Processing, Englewood Cliffs, PrenticeHall.

[7] U. Beis. An Introduction to Delta Sigma Converters [Online]. Available: http://www.beis.de/Elektronik/DeltaSigma/DeltaSigma.html

[8] B. Baker. (2011). How delta-sigma ADCs work [Online]. Available: http://www.ti.com/general/docs/lit/getliterature.tsp?baseLiteratureNumber=slyt423

[9] W. Kester. (2004). Analog-Digital Conversion, chapter 6. ISBN 978-0916550271

[10] Phillips Semiconductors. (1986). I2S bus specifications [Online]. Available: https://sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf

[11] Texas Instruments. (December 2014). TLV320AIC3101 Low-Power Stereo Audio Codec for Portable Audio/Telephony [Online]. Available: http://www.ti.com/lit/ds/symlink/tlv320aic3101.pdf

[12] M. Wright. (2013, Feb, 20). Implementing USB 3.0 in MCU- and Microprocessor-Based Systems [Online]. Available: http://www.digikey.com/en/articles/techzone/2013/feb/implementing-usb-30-in-mcu-and-microprocessorbased-systems

[13] Microsoft Corporation. (2015). How to transfer data to USB isochronous endpoints [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/hardware/hh406225(v=vs.85).aspx

[14] J. Axelson. (2005). USB Complete: Everything You Need to Develop Custom USB Peripherals, Chapter 3: A Transfer Type for Every Purpose.

[15] Tech Design Forums - USB 3.0 protocol layer [Online]. (2015, Apr). Available: http://www.techdesignforums.com/practice/technique/usb-3-0-protocol-layer-2/

[16] Cypress Semiconductor Corporation. (2010, Feb). USB 3.0 - The Next-Generation Interconnect [Online]. Available:
http://www.cypress.com/?docID=21349

[17] Inno-Logic, Inc. (2015). USB 3.0 SuperSpeed Overview [Online]. Available:
http://inno-logic.com/usb-3-superspeed-overview.php

[18] Linux Kernel Organization, Inc. USB Documentation [Online]. Available:
https://www.kernel.org/doc/Documentation/usb/bulk-streams.txt

[19] Sara Shakil Qureshi. (2014). Bulk transfer - Comparison of USB 2.0 & USB 3.0 [Online]. Available:
http://asicfpga.seecs.nust.edu.pk/downloads/BULK%20TRANSFER.pdf

[20] USB Implementers Forum, Inc. (2009, May, 20-21) SuperSpeed USB Developers Conference - Isochronous Protocol [Online]. Available:
http://www.usb.org/developers/presentations/SuperSpeed_USB_DevCon_Isochronous_Froelich.pdf

[21] J. Donovan. (2011 March 6). USB 3.0 - Are We There Yet? [Online]. Available:
http://www.digikey.com/en/articles/techzone/2011/apr/usb-30--are-we-there-yet

[22] S. Smith. (1997). The Scientist and Engineer's Guide to Digital Signal Processing. 1st ed. San Diego: California Technical processing, pp. 35-66.

[23] Adrian Nastase. (2011). Design a Unipolar to Bipolar Converter for a Unipolar Voltage Output DAC [Online]. Mastering Electronic Design. Available:
http://masteringelectronicsdesign.com/design-a-unipolar-to-bipolar-converter-for-a-unipolar-voltage-output-dac/

[24] E. Balestrieri and S. Rapuano. (2006, Dec). Defining DAC performance in the frequency domain. Measurement [Online]. 40, pp 463-472. Available:
http://www.elsevier.com/locate/measurement.

[25] IEEE Std. (2001). IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters, IEEE, ISBN 0-7381-2724-8.

[26] Walter Kesler. (2009). Understand SINAD, ENOB, SNR, THD, THD + N, and SFDR so You Don't Get Lost in the Noise Floor [online]. Available:
http://www.analog.com/media/en/training-seminars/tutorials/MT-003.pdf

[27] Cypress Semiconductor Corporation. (2012, June, 26). GPIF™ II Designer [Online]. Available:
http://www.cypress.com/?rID=59628

[28] Cypress Semiconductor Corporation. (2013, July, 9). GPIF Designer [Online]. Available:
http://www.cypress.com/?rID=14448

[29] R. Krishna (2013). Designing a GPIF™ II Master Interface [Online]. Available:
http://www.cypress.com/?docID=47100

[30] Cypress Semiconductor Corporation. (2015, Jan, 7). EZ-USB FX3S SuperSpeed USB Controller [Online]. Available:
http://www.cypress.com/?docID=45991

[31] Cirrus Logic, Inc. (2008, May). CS4384 Datasheet [Online]. Available:
http://www.cirrus.com/en/pubs/proDatasheet/CS4384_F1.pdf

[32] Walter Kester. (2009). Oversampling interpolation filters [Online]. Available:
http://www.analog.com/media/cn/training-seminars/tutorials/MT-017.pdf

[33] Texas instruments (2006). TL97x Output Rail-To-Rail Very-Low-Noise Operational Amplifiers. Available:
http://www.ti.com/lit/ds/symlink/tl971.pdf

[34] Universal Serial Bus 3.0 Specification (datasheet), page 11-6 [Online]. Available:
http://www.usb3.com/whitepapers/USB%203%200%20(11132008)-final.pdf

[35] Linear Technology (2013). Basic Concepts of Linear Regulator and Switching Mode Power Supplies [Online]. Available:
http://cds.linear.com/docs/en/application-note/AN140fa.pdf

[36] Murata Power Solutions, Inc. NTA Series. Available:
http://www.mouser.com/ds/2/281/kdc_nta-50920.pdf

[37] Cypress Semiconductor, Inc. EZ-USB® FX3™ Development Kit Guide [Online]. Available:
http://www.element14.com/community/docs/DOC-54485/l/cypress-user-guide-for-cyusb3kit-001-ez-usb-fx3-development-kit

[38] Texas Instruments. (2008). PCM1690 24-Bit, 192-kHz Sampling, Enhanced Multi-Level ΔΣ, Eight-Channel Audio Digital-to-Analog Converter [Online]. Available:
http://www.ti.com/lit/ds/symlink/pcm1690.pdf

[39] element14. (2014, Feb). Cypress: CYUSB3KIT-001 EZ-USB® FX3™ Development Kit [Online]. Available:
http://www.element14.com/community/docs/DOC-54488/l/cypress-cyusb3kit-001-ez-usb-fx3-development-kit

[40] Y.T.B. Mulder, J. Romijn, D.S.M. Verhaert. (2015). A High-Speed 40-channel USB 2.0 DAC for Adaptive Optics. ISBN 978-94-6186-493-2