

Dokumentation

Radike, Sauer, Wolf

April 15, 2021

Version	Datum	Autor	Änderungsgrund/Bemerkungen
1.0	11.02.2021	Sauer Lukas	Erstellung
1.1	25.02.2021	Sauer Lukas	intADC
1.3	18.03.2021	Sauer Lukas	Vorwort & ADC-PCB & ext. ADC
2.0	08.04.2021	Sauer Lukas	Kommunikationsprotokoll
2.1	15.04.2021	Sauer Lukas	update Kommunikationsprotokoll

Contents

1	Vorwort	4
1.1	Projektübersicht	4
1.2	Projektgruppe	4
1.3	Projektbetreuer	5
2	Entwicklung	5
2.1	Top-Level-Design-Unit Grundstruktur	5
2.1.1	Projektteilbereich Übersicht	5
2.2	Verwendung der internen ADCs	5
2.2.1	Projektteilbereich Übersicht	5
2.2.2	ADC Hardware	5
2.2.3	VHDL Implementierung	6
2.3	ADC-PCB "Prototype 1"	7
2.3.1	Projektteilbereich Übersicht	7
2.3.2	Schaltplan	8
2.3.3	PCB	8
2.4	externer ADC	12
2.4.1	Projektteilbereich Übersicht	13
2.4.2	LTC1420C technische Merkmale	13
2.4.3	VHDL Implementierung	13
2.4.4	erste Testversuche	13
2.5	Trigger	13
2.5.1	Projektteilbereich Übersicht	13
2.5.2	Funktion	13
2.5.3	VHDL-Design-Unit	16
2.5.4	Design-Unit Test	16
2.6	Kommunikationsprotokoll	16
2.6.1	Projektteilbereich Übersicht	16
2.6.2	Paketspezifikation FPGA zu UserInterface	16
2.6.3	Paketspezifikation UserInterface zu FPGA	16
2.6.4	Messbereichseinstellung	16
2.6.5	Zeitbereichseinstellung	17
2.6.6	Messdaten	17
2.6.7	Checksum	17
2.6.8	frei / not used	18
3	verwendete Messgeräte & Entwicklungsboards	18
3.1	DE10-Lite Board	18
3.2	Oszilloskop 1	18


1 Vorwort

1.1 Projektübersicht

Im Projekt Oszi wird ein Oszilloskop mittels eines FPGAs zu realisieren. Das Oszilloskop soll ein Frequenzband von 0Hz bis 500kHz einlesen und einen Spannungsbereich von -20V bis +20V abdecken. Die Spannungskurve soll über ein Computerprogramm graphisch ausgegeben werden. Über das Programm soll ebenfalls der Spannungs- und Zeitbereich der Ausgabe einzustellen sein. Das Oszilloskop solle triggerbar sein und Tastköpfe kalibrieren können. Das Projekt besteht aus drei Aufgabenbereichen, dem Analog-Front-End, der Datenverarbeitung mit dem FPGA und der Ausgabe am PC.

1.2 Projektgruppe

Das Team entstand im Zuge des Sommersemester-Projekts in der 4. Schulstufe am TGM. Gleiche Interessen, gutes technisches Wissen und Verständnis, sowie eine gute Harmonie unter den Gruppenmitgliedern führte zum Zusammenschluss. Die Projektgruppe ist geschlossen aus einer Klasse, der 4AHEL 2021 am TGM.

Projektteilnehmer		
Radike Markus	Sauer Sauer	Wolf Benedict
Bild 1		bild 3
user interface	digital data processing	analog front end

1.3 Projektbetreuer

Fachbereich	Lehrperson
Labor	GRÄBNER Kar-Heinz
Werkstatt	GRAUPE Andreas

2 Entwicklung

2.1 Top-Level-Design-Unit Grundstruktur

2.1.1 Projektteilbereich Übersicht

2.2 Verwendung der internen ADCs

2.2.1 Projektteilbereich Übersicht

Die im DE10-Lite Board integrierten ADCs werden versucht am Anfang des Projektes anzusteuern und vorläufig, bevor ein Piggyback mit einem externen ADC vorhanden ist, zu verwenden. **Dieser erste Schritt ist fehlgeschlagen, da die Verwendung der ADCs nur mit einem eingebetteten Nios-Prozessor möglich ist.** Die Implementierung eines solchen komplexen Bausteins würde die zeitlichen Grenzen maßlos überschreiten und ist auch nicht Ziel des Projekts. Die geleistete Arbeit ist trotzdem dokumentiert und auf den folgenden Seiten enthalten.

Das Board hat 6 integrierte ADCs. Diese sind mit einer Bandbreite von 12 Bit und einer Sampling-Frequenz von 10 MHz spezifiziert und haben eine Verstärkerschaltung vorgeschaltet. Eine analoge Eingangsspannung von 0-5V ist zulässig. Quartus-Prime bietet bereits fertige IP-Cores zur erleichterten Implementierung an.

2.2.2 ADC Hardware

Die auf dem Chip *MAX 10 10M50DAF484C7G* realisierten 6 ADCs werden verwendet, so ist kein extra Chip für die Analog-Digital-Wandlung notwendig. Diese ADCs haben eine Auflösung von 12 Bit, eine ADC Clock-frequency von 10MHz und verwenden die interne Referenzspannung von 2,5V. Herausgeführt sind die ADC-Pins **indirekt** an den sogenannten Arduino-Pins, wie in Abbildung 1 zu sehen. Das "analog Front-End" enthält einen Vorverstärker, realisiert mittels OPV, der grundsätzlich die Aufgabe hat die Spannung zu halbieren. Somit werden die maximal zulässigen 5V auf 2,5V heruntergeregelt, welche die Referenzspannung des ADCs ist und diesen so nicht übersteuert. Die Transitfrequenz des MCP6244 liegt bei 550kHz. Bei 10MHz

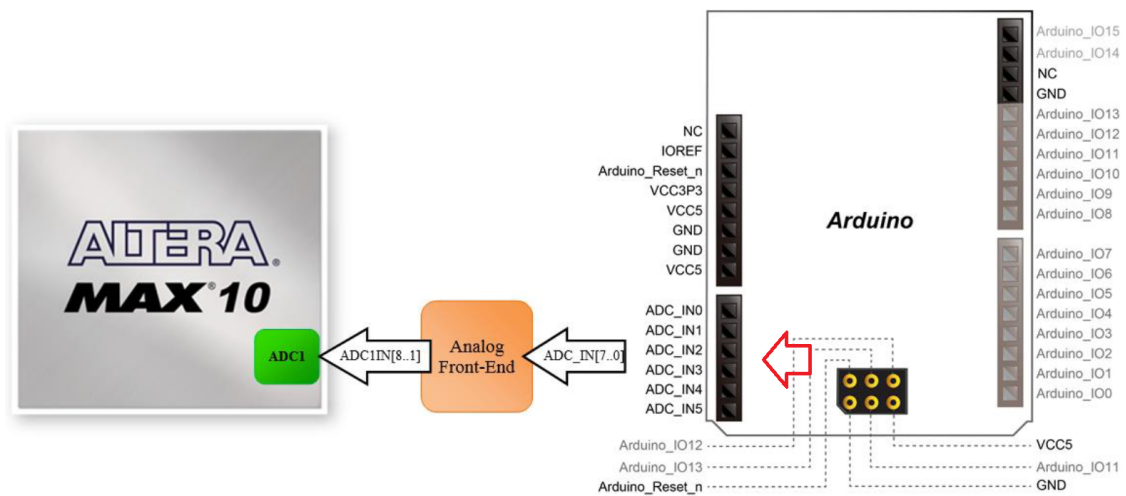


Figure 1: Blockschaltbild

geht sich eine Verstärkung von 0,5 knapp aus, was jedoch wenig Bedeutung hat, da die Frequenz der Eingangssignale, aufgrund der Abtastrate von 10MSa/s, ohnehin kleiner sein muss.

Wie in dem Schaltplanausschnitt (aus FPGA-Schaltplan: 4) in Abbildung 2 zu sehen, sind auch Kondensatoren vorhanden. Das Verhalten im Frequenzbereich wurde mithilfe von LtSpice analysiert. Da der originale OPV im Simulationsprogramm nicht spezifiziert ist, wurde der sehr ähnliche AD8038 verwendet.

Im Bodediagramm (Figure: 3) ist die Verstärkung von $\frac{1}{2} = -6dB$ gut zu erkennen. Der Abfall von -20 dB pro Dekade beweist die 1. Ordnung. Die Grenzfrequenz (*hierbei* - 9dB) liegt bei 997kHz. Das ist locker ausreichend, da die Frequenz der Eingangssignale aufgrund der Abtastrate von 10MSa/s sowieso niedriger ist.

2.2.3 VHDL Implementierung

Es wurde mit verschiedenen IP-Cores von Quartus versucht die ADCs anzusteuern. Eine Implementierung mit dem IP-Core "Modular ADC core Intel FPGA IP" über das "Avalon"-Interface wurde probiert, schlug jedoch fehl, da der "Fitter" von Quartus Prime nicht für eine Ansteuerung ohne Nios-Prozessor ausgelegt ist. **Fazit:** Die internen ADCs im DE10-Lite Board können nur mit einem eingebetteten Nios Prozessor verwendet werden.

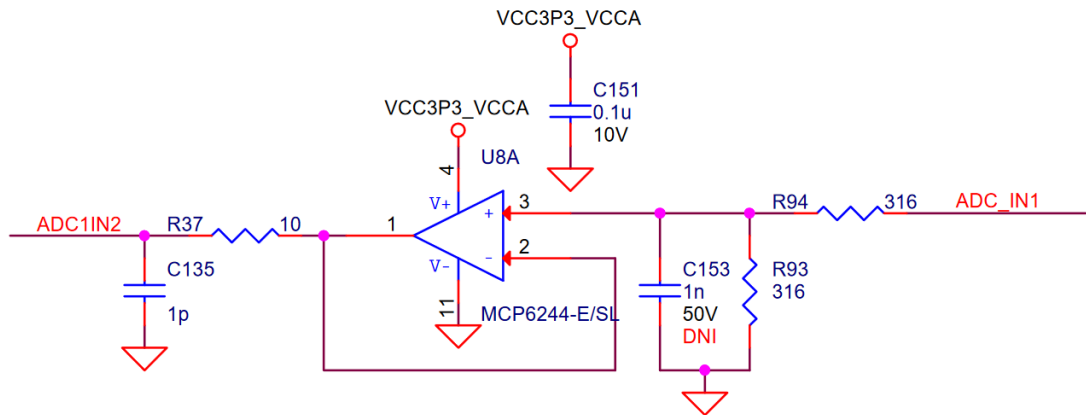


Figure 2: analog Front-End Ch1

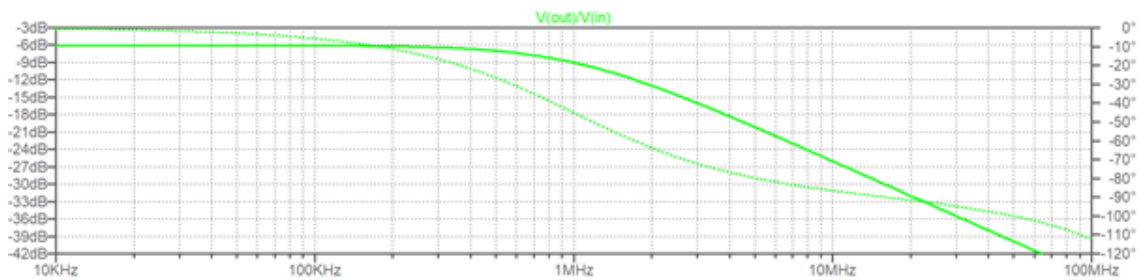


Figure 3: Bodediagramm Vorverstärker

2.3 ADC-PCB "Prototype 1"

2.3.1 Projektteilkbereich Übersicht

Zum Testen und erstmaligen Ansteuern des externen ADCs (siehe: 2.4) wurde ein extra PCB angefertigt. Der Schaltplan und das Platinenlayout wurden in KiCad gezeichnet, die Gerberdateien exportiert und die Platine anschließend mithilfe einer CNC-Fräsmaschine gefertigt. Das Löten von sehr kleinen SMD-Bauteilen war neu für uns und wir freuten uns das dazuzulernen. Unterstützung bekam die Projektgruppe von Prof. Graupe.

2.3.2 Schaltplan

Der Schaltplan (Figure: 4) wurde möglichst simpel gehalten, es befindet sich ausschließlich der ADC mit seinen obligatorischen Stützkondensatoren auf der Platine und Pinheader zum Anschließen, sowie ein Eingangswiderstand für den Analogeingang. Die Spannungsversorgung als auch die Kommunikation erfolgen über die Pinheader zum FPGA. Der Verwendete ADC ist ein LTC1420C (siehe: 2.4.2).

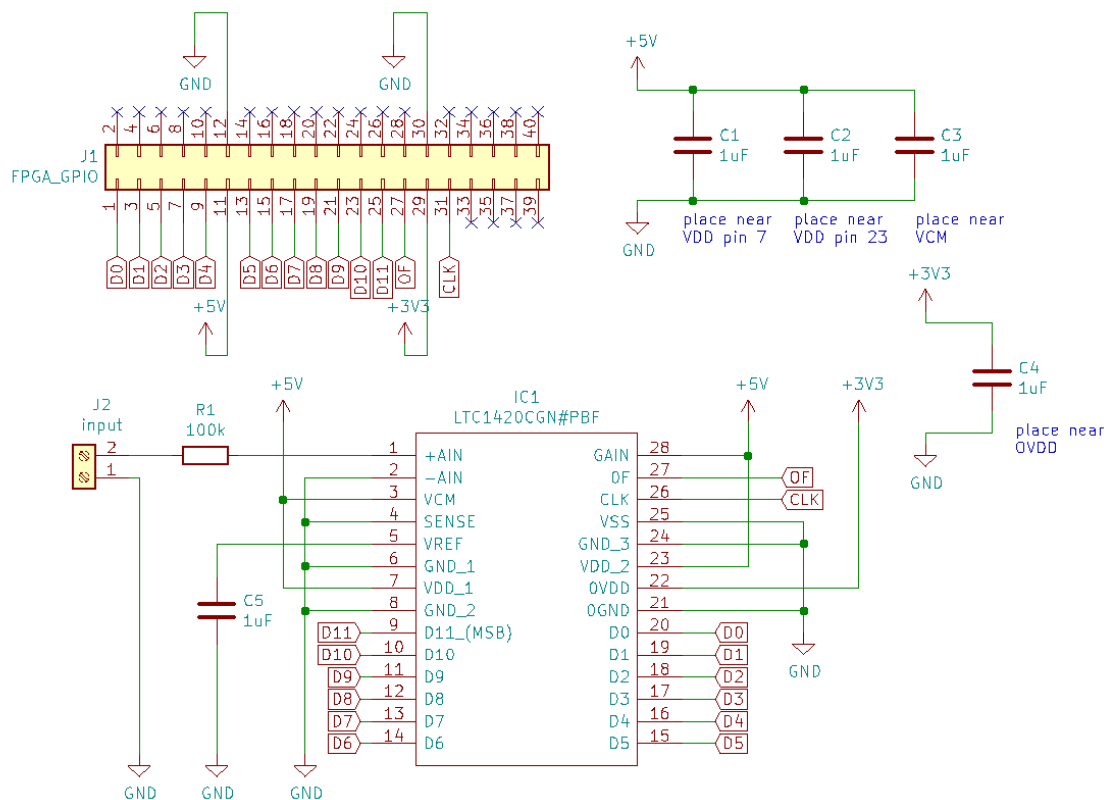


Figure 4: Schaltplan "Prototype 1"

2.3.3 PCB

Das Platinenlayout ist einseitig ausgeführt, möglichst simpel und klein gehalten. Über 2x20 Pinheader wird das Board auf das FPGA gesteckt und sowohl elektrisch als auch mechanisch verbunden. Die Platine hat die Maße 54,6 x 23,5 mm, wie in Figure 10 ersichtlich. Die Nähe der Clock-Leitung zu

der Leitung für das analoge Eingangssignal ist sehr unvorteilhaft, jedoch für die Kommunikation zwischen FPGA und ADC irrelevant.

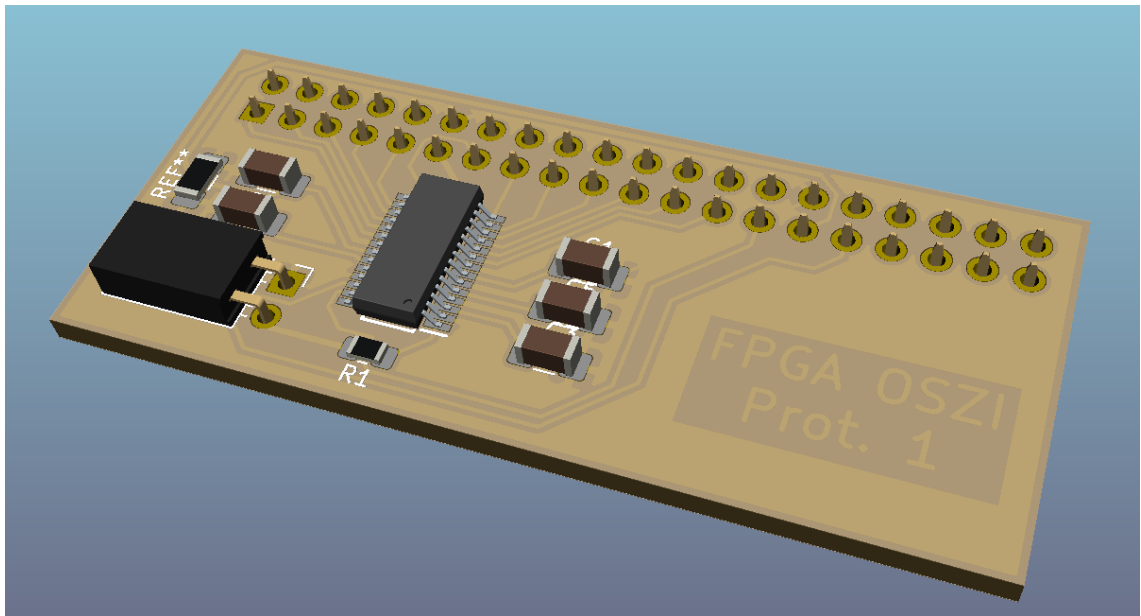


Figure 5: 3D-Ansicht Vorderseite

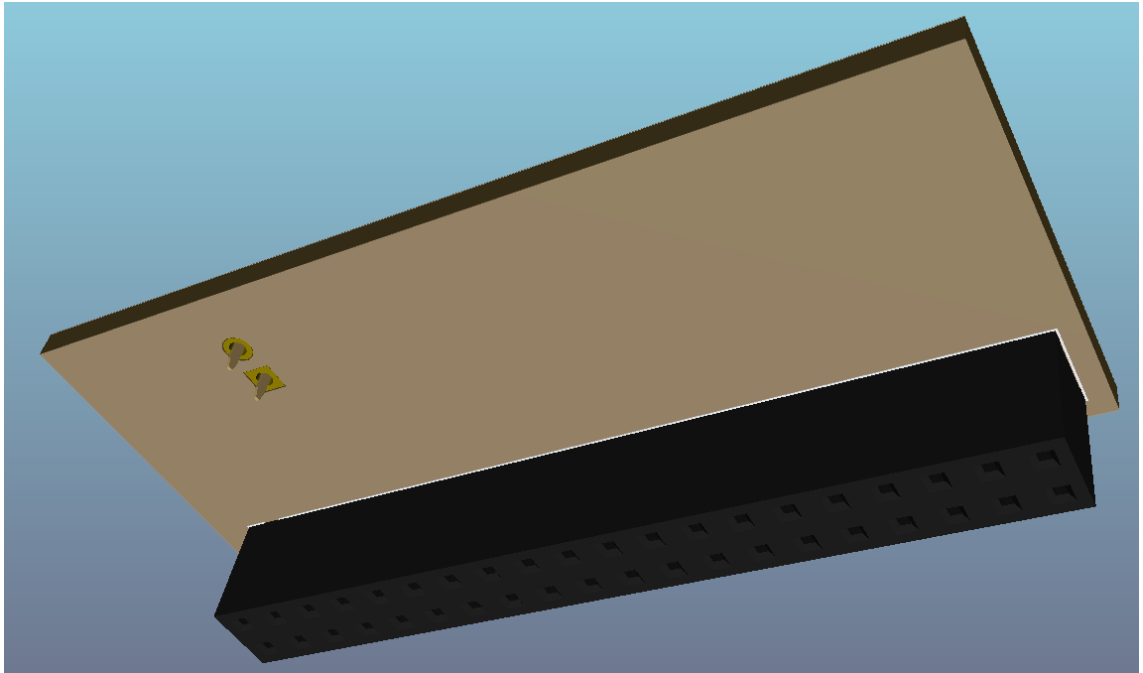


Figure 6: 3D-Ansicht Rückseite

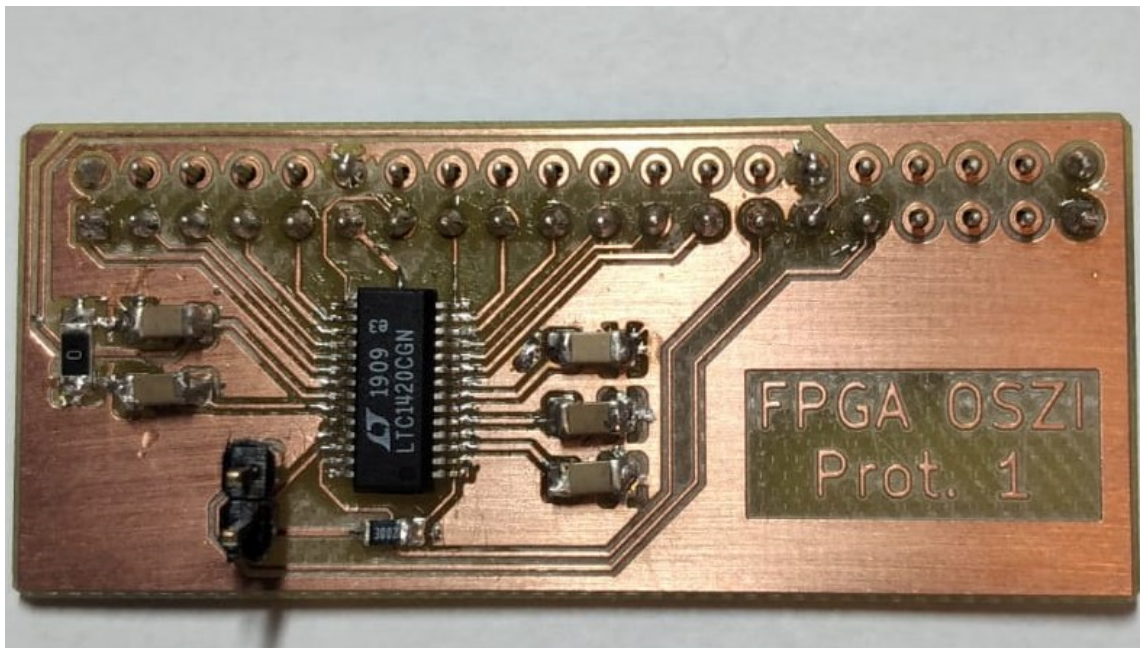


Figure 7: Fotografie fertige Platine

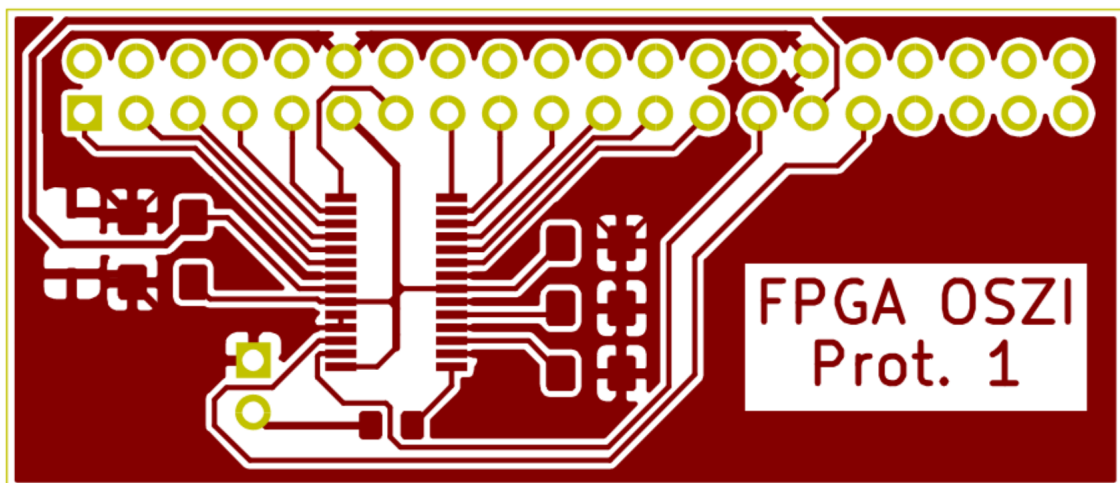


Figure 8: Kupferlayout Vorderseite

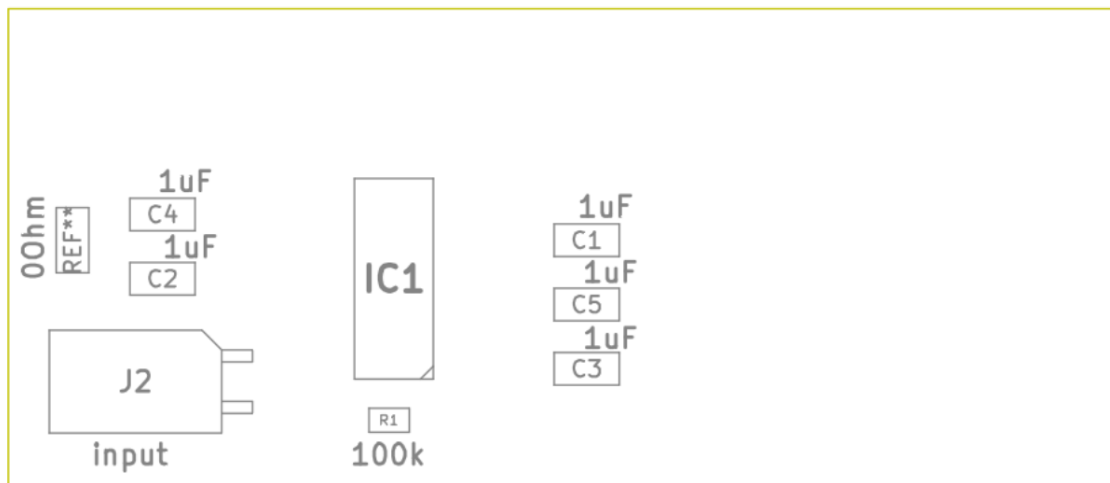


Figure 9: Bestückungsplan Vorderseite

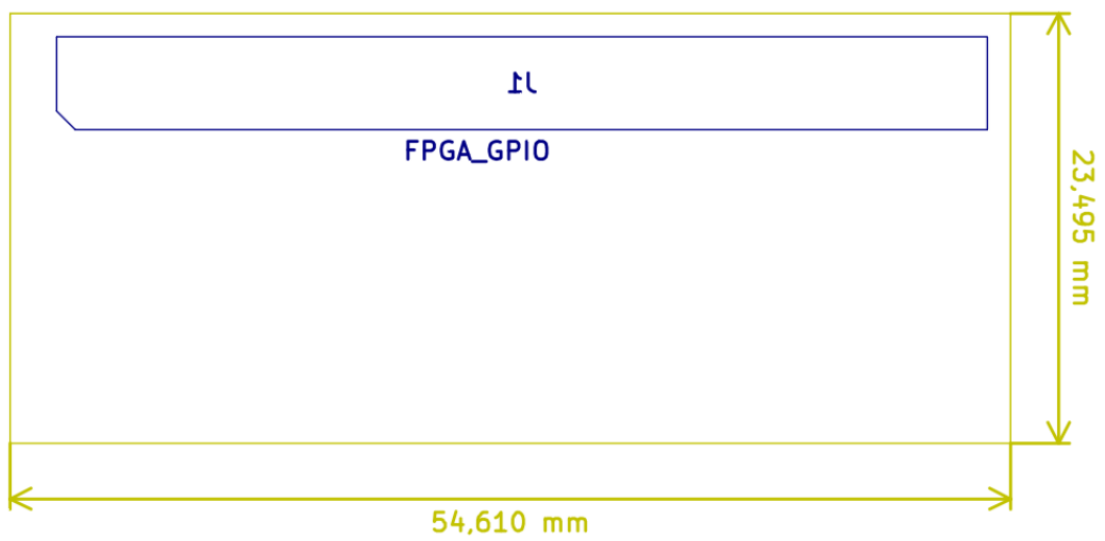


Figure 10: Bestückungsplan Rückseite & Bemaßung

2.4 externer ADC

2.4.1 Projektteilbereich Übersicht

Die Wahl des externen ADCs ist auf den LTC1420C gefallen. Dieser besitzt eine Auflösung von 12-Bit bei einer Abtastrate von 10MSa/s. Ausgelesen werden die Messwerte vom FPGA parallel über 12 digitale Ausgänge. Der ADC benötigt keine Referenzspannung, da diese intern erzeugt wird, jedoch eine Clock von 10Mhz muss bereit gestellt werden.

2.4.2 LTC1420C technische Merkmale

Das Datenblatt wurde unter den Quellen verlinkt, siehe 4

2.4.3 VHDL Implementierung

2.4.4 erste Testversuche

Für die ersten Testversuche wurde das *Oszilloskop 1 (3.2)* verwendet.

2.5 Trigger

2.5.1 Projektteilbereich Übersicht

Der Edge-Trigger ist wesentlich für die Unterteilung in Pakete. Dieser reagiert auf eine vorgegebene, konfigurierbare Schwellspannung in jedem der drei Modi. Der gewünschte Modus wird vom User-Interface vorgegeben. Einen sogenannten "Single-Shot" kennt diese Komponente nicht, da diese ausschließlich primitiv triggert und sich nicht um "höhere Angelegenheiten" kümmert.

2.5.2 Funktion

Modus: Rising Edge Bei der *Rising-Edge-Detection* löst der Trigger beim Überschreiten des Schwellwertes (in der Grafik: **rot**) aus. Es muss jedoch immer davor, aufgrund der Hysterese, ein vom Schwellwert abhängiger unterer Wert (in der Grafik: **blau**) unterschritten werden (`arming_level_low`). Damit wird falsches, zu häufiges Auslösen wegen höherfrequenten und/oder überlagerten Signalen vermieden. Signale mit einer zu kleinen Amplitude (\leq Hysterese) können nicht getriggert werden. In der Grafik 11 sind 3 Szenarien Abgebildet, nur im ersten löst der Trigger aus.

Modus: Falling Edge Bei der *Falling-Edge-Detection* löst der Trigger beim Unterschreiten des Schwellwertes (in der Grafik: **rot**) aus. Es muss jedoch immer davor, aufgrund der Hysterese, ein vom Schwellwert abhängiger oberer Wert (in der Grafik: **blau**) überschritten werden (`arming_level_high`).

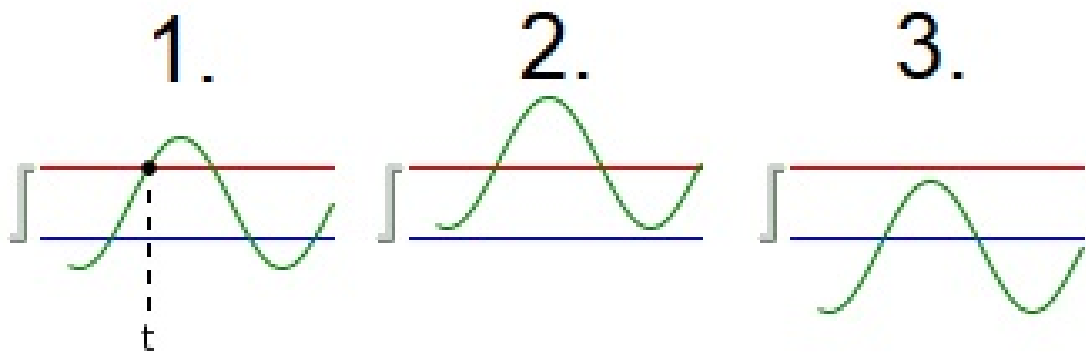


Figure 11: Trigger Rising-Edge-Detection

Damit wird falsches, zu häufiges Auslösen wegen höherfrequenten und/oder überlagerten Signalen vermieden. Signale mit einer zu kleinen Amplitude (\leq Hysterese) können nicht getriggert werden. In der Grafik 12 sind 3 Szenarien Abgebildet, nur im ersten löst der Trigger aus.

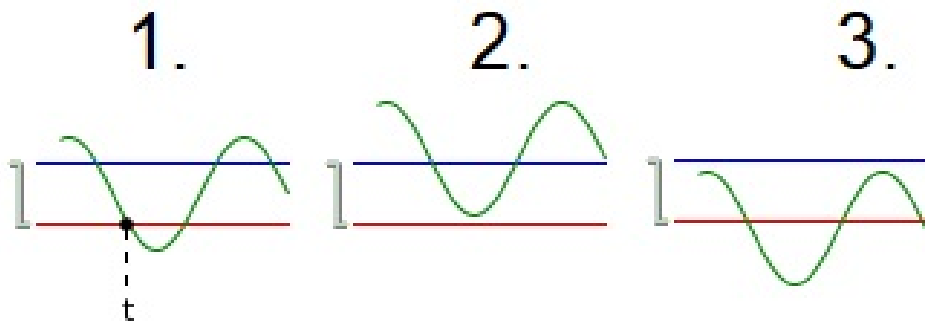


Figure 12: Trigger Falling-Edge-Detection

Modus: Any Edge Bei der *Any-Edge-Detection* löst der Trigger sowohl beim Unterschreiten als auch beim Überschreiten des Schwellwertes (in der Grafik: **rot**) aus. Es muss jedoch immer davor, aufgrund der Hysterese, ein vom Schwellwert abhängiger oberer oder unterer Wert (in der Grafik:

blau) überschritten bzw. unterschritten werden (`arming_level_high`, `arming_level_low`). Damit wird falsches, zu häufiges Auslösen wegen höherfrequenten und/oder überlagerten Signalen vermieden. Signale mit einer zu kleinen Amplitude (\leq Hysterese) können nicht getriggert werden. In der Grafik 13 sind 4 Szenarien Abgebildet, in der ersten Zeile funktioniert das Triggern, in der zweiten nicht.

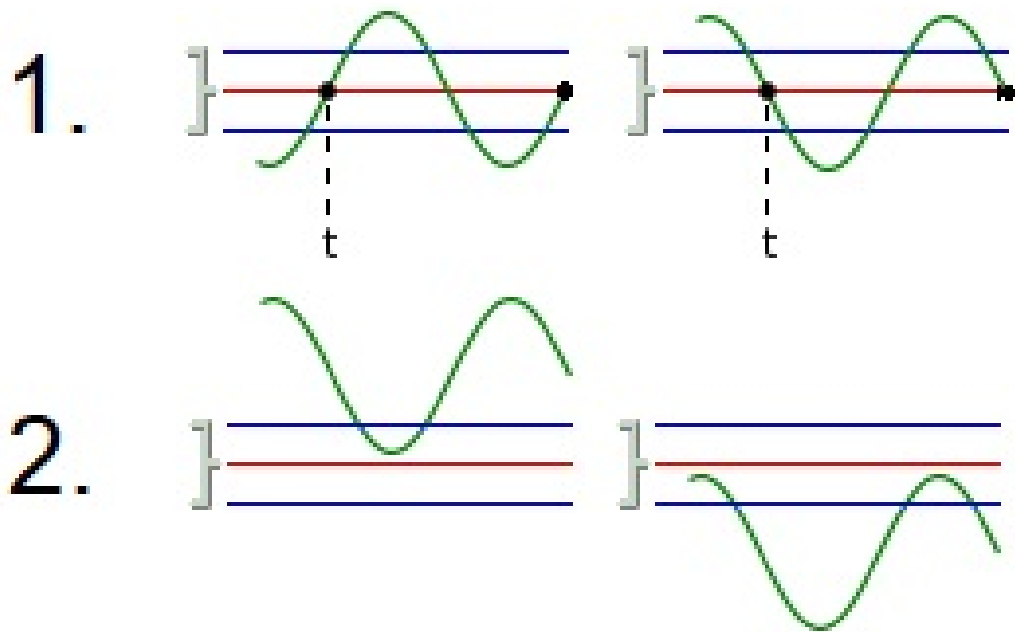


Figure 13: Trigger Any-Edge-Detection

Schwellwert, Hysterese und Modi Diese Konfigurationseinstellungen werden vom User-Interface vorgegeben. (siehe: 2.6) Für den Schwellwert kann jeder Wert (\approx Spannung) eingestellt werden, jedoch sollte darauf geachtet werden, dass die Hysterese im aktuellen Modus den Wertebereich nicht verlässt. Die Trigger-Unit begrenzt sicherheitshalber die Hysterese dynamisch so, dass diese den Wertebereich nicht verlassen kann. ($arming_level_low \geq 0$, $arming_level_high \leq 4095$) Der Schwellwert wird nicht in Volt angegeben, sondern im Bereich der Rohdaten des ADCs (0 - 4095). Eine Hysterese im Bereich von 20 bis 100 digits wird empfohlen.

2.5.3 VHDL-Design-Unit

Die Komponente ist sehr einfach gehalten und auch zu bedienen bzw. zu

implementieren.

Port	Typ	I/O
RESET_n	std_logic	in
CLK	std_logic	in
values_in_i	natural	in
trigger_threshold_i	natural	in
trigger_hyst_i	natural	in
trigger_mode	trigger_types	in
trigger	std_logic	out

2.5.4 Design-Unit Test

2.6 Kommunikationsprotokoll

2.6.1 Projektteilbereich Übersicht

Für die Kommunikation zwischen dem FPGA und dem UserInterface wurde ein spezielles Protokoll entwickelt, welches auf UART aufbaut. Ziel ist ein schnelles Senden der Messdaten und Austauschen der Konfiguration mit inkludierter Fehlerdetektion. Das gesamte Protokoll ist aufgebaut aus einzelnen Daten mit einer Wortlänge von 12-Bit, was die Handhabung der 12-Bit-ADC-Messwerte vereinfacht und praktisch für größere Zahlen ist. Dass mit UART auf unterer Ebene jeweils 8 Bit gesendet werden, wird ignoriert und ist für die höheren Ebenen irrelevant. Es werden immer Pakete mit einem identischen Aufbau gesendet.

2.6.2 Paketspezifikation FPGA zu UserInterface

2.6.3 Paketspezifikation UserInterface zu FPGA

2.6.4 Messbereichseinstellung

Der Messbereich wird über diesen Wert eingestellt. Das FPGA-Oszilloskop bekommt diesen vom UserInterface vorgegeben und schickt diesen zur Kontrolle mit den Messwerten auch wieder zurück. In der folgenden Tabelle sind

	Wert / Nummer	Messbereich
die durchnummerierten Messbereiche zu finden.	0	1 : 1 (beispiel)
	1	1 : 10 (beispiel)
	2	1 : 24 (beispiel)

2.6.5 Zeitbereichseinstellung

Die Zeitbereichseinstellung bezieht sich auf einen Zeitabschnitt (Division). Davon sind immer zehn in der Grafik im UserInterface zu sehen, ein Raster sozusagen. So wie bei den alten, analogen Oszilloskopen, bei denen auch ein Raster am Bildschirm zu sehen war. Hat ein Zeitabschnitt beispielsweise eine Dauer von 20ms, so sind in der Anzeige insgesamt 200ms abgebildet.

Um die große, unilineare Spanne von Zeitbereichen mit 12 Bit abzudecken, hat die Projektgruppe hier eine Wertetabelle geplant.

Wert / Nummer	Zeitbereich	Wert / Nummer	Zeitbereich
0	2 μ s	12	5 ms
1	5 μ s	13	10 ms
2	10 μ s	14	20 ms
3	10 μ s	15	50 ms
4	20 μ s	16	100 ms
5	50 μ s	17	200 ms
6	100 μ s	18	500 ms
7	200 μ s	19	1 s
8	500 μ s	20	2 s
9	1 ms	21	5 s
10	2 ms	22	10 s
11	5 ms		

2.6.6 Messdaten

Die Samples werden direkt aus dem Buffer als Bit-Stream zum UserInterface geschickt. Dieses trennt die Samples anschließend wieder durch zählen, alle 12 Bit beginnt ein neuer Messwert. Es werden immer genau 1200 Messwerte chronologisch (frühester Messwert zuerst, MSB voran)geschickt, da eine Genauigkeit von 20 Samples pro Zeitabschnitt festgelegt ist und immer 60 Zeitabschnitte gesendet werden. 60 mal 20 Samples mal 12 Bit ergibt eine Gesamtanzahl von 14400 Bits.

2.6.7 Checksum

Die sogenannte "Checksum" ist die Quersumme des gesamten, bisher gesendeten Datenpakets. Alle logischen Einser werden addiert und als Checksum mitgesendet. So können Übertragungsfehler mit hoher Wahrscheinlichkeit festgestellt werden. Die Quersumme kann vom UserInterface zum FPGA maximal 240 betragen und umgekehrt maximal 14544. Beide der Kontrollzahlen setzten sich aus 2 Wortlängen zusammen, somit 24 Bit.

2.6.8 frei / not used

Diese Abschnitte werden (noch) nicht benutzt und dienen für spätere, eventuelle Erweiterungen. Eine zusätzliche Spektralanalyse-Funktion könnte die freien Wörter nutzen.

3 verwendete Messgeräte & Entwicklungsboards

3.1 DE10-Lite Board

Seriennummer und Produktnummer ect. + Foto

3.2 Oszilloskop 1

Art	Oszilloskop	Foto
Produktnummer	HM1507-2	
Seriennummer	/	
TGM Inv. Nr.	540-16/22/99	
Beschreibung	digitales Oszilloskop 2 Channel 150 MHz / 200 MSa/s Röhrenbildschirm	

4 Quellen und Hilfsmittel

Verwendung des DE10-Lite-Boards & VHDL-Programmierung:

- Übersicht: DE10-Lite-Board: DE10-Lite_v.2.1.0_SystemCD: DE10-Lite_User_Manual.pdf
- Schaltplan: DE10-Lite-Board: DE10-Lite_v.2.1.0_SystemCD: de10-lite.pdf
- VHDL-Spezifikation: <https://www.nandland.com>
- Infos zum Triggern: <https://www.tiepie.com/en/fut/edge-trigger>

Datenblätter:

- **LTC1420C**: <https://www.analog.com/media/en/technical-documentation/data-sheets/1420fa.pdf>



SAUER/Grafiken/Trigger/EntityTrigger.png

Figure 14: Entity Trigger

Datenpaket/Protokoll vom FPGA-Oszi zum User-Interface:

1	1	10	1200	2	words
12	12	120	14400	24	Bits
Messbereich	Zeithereich	frei / not used	Datenstream 60 mal 20 = 1200 12-Bit-Smaples	Checksum Quersumme	

words insgesamt: 1214 bits insgesamt: 1608

Figure 15: Paketspezifikation FPGA zu UserInterface

Datenpaket/Protokoll vom User-Interface zum FPGA-Oszi:

1	1	1	1	112	15	2	words
12	12	12	12	180	12		Bits
Messbereich	Zeithereich	Triggermode	Triggerwert	Triggerhysteresese vorläufig konst.	frei / not used	Checksum Quersumme	

words insgesamt: 22 bits insgesamt: 264

Figure 16: Paketspezifikation UserInterface zu FPGA