

Dokumentation

Radike, Sauer, Wolf

April 29, 2021

Version	Datum	Autor	Änderung/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
3.0	15.04.2021	Sauer Lukas	Trigger

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 Der ADC - LTC1420C	7
2.3.1 VHDL "ADC Interfcae"	7
2.4 ADC-PCB "Prototype 1"	8
2.4.1 Projektteilbereich Übersicht	8
2.4.2 Schaltplan	8
2.4.3 PCB	9
2.5 Trigger	11
2.5.1 Projektteilbereich Übersicht	11
2.5.2 Funktion	11
2.5.3 VHDL-Design-Unit	14
2.5.4 Design-Unit Simulation	14
2.5.5 Trigger Praxistest	15
2.6 Kommunikationsprotokoll	16
2.6.1 Projektteilbereich Übersicht	16
2.6.2 Paketspezifikation FPGA zu UserInterface	17
2.6.3 Paketspezifikation UserInterface zu FPGA	17
2.6.4 Messbereichseinstellung	17
2.6.5 Zeitbereichseinstellung	18
2.6.6 Messdaten	19
2.6.7 Checksum	19
2.6.8 frei / not used	19
2.7 UART - Verbindung FPGA & UserInterface	20
2.7.1 Projektteilbereich Übersicht	20
2.7.2 erste Verbindung	20

3	verwendete Messgeräte & Entwicklungstools	21
3.1	DE10-Lite Board	21
3.2	Oszilloskop 1	22
3.3	Funktionsgenerator 1	23
3.4	RS232-Adapter	24
4	Quellen und Hilfsmittel	24

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
		
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 vorgeschalten. 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 heruntergeregt, welche die Referenzspannung des ADCs ist und diesen so nicht übersteuert. Die Transitfrequenz des MCP6244 liegt bei 550kHz. Bei 10MHz

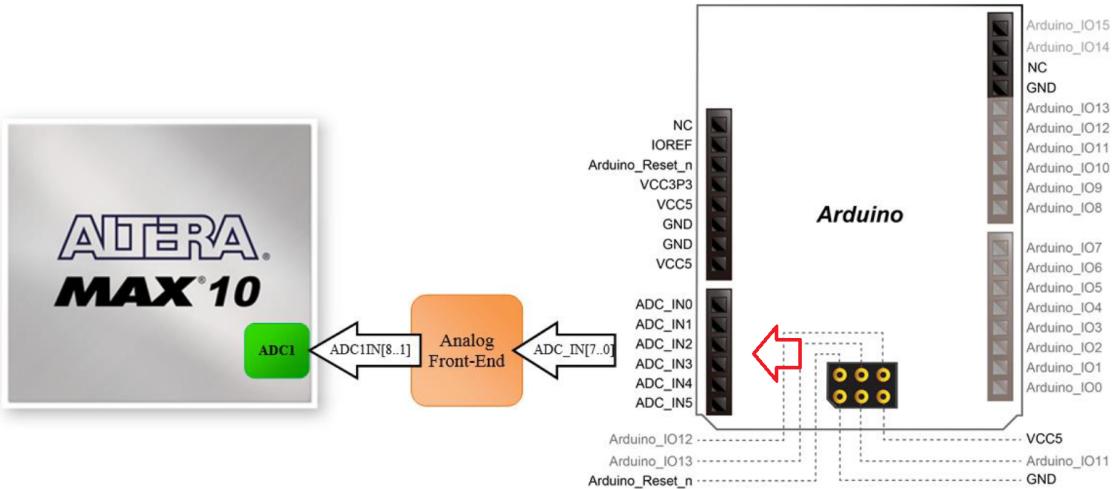


Figure 1: Blockschaltbild

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 orginale 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.

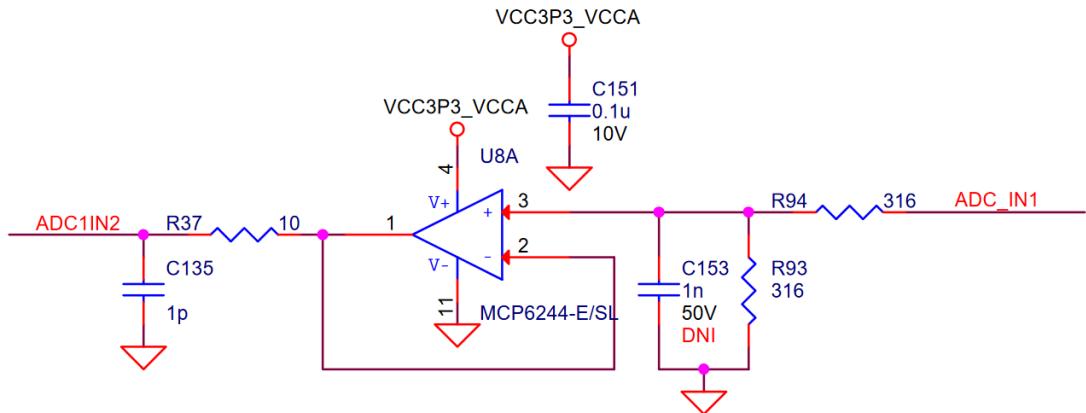


Figure 2: analog Front-End Ch1

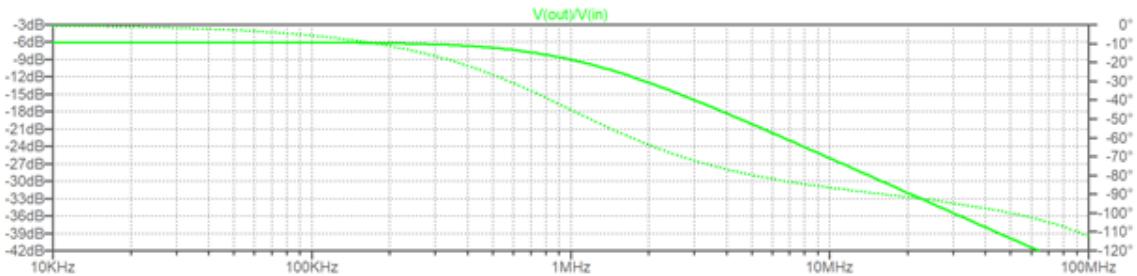


Figure 3: Bodediagramm Vorverstärker

2.3 Der ADC - LTC1420C

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 Referezspannung, da diese intern erzeugt wird, jedoch eine Clock von 10Mhz muss bereit gestellt werden. Das Datenblatt ist in den Quellen verlinkt, siehe: 4

2.3.1 VHDL "ADC Interfcae"

Für die Kommunikation mit dem externen ADC wurde eine eigene VHDL-Design-Unit *ADCinterface* geschrieben.

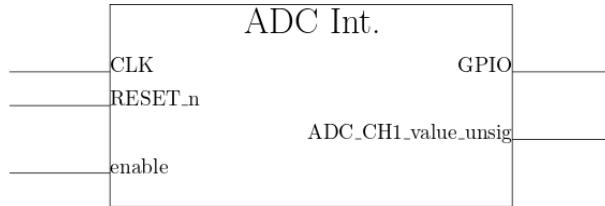


Figure 4: Entity ADC Interface

Port	Typ	I/O
RESET.n	std_logic	in
CLK	std_logic	in
GPIO	std_logic_vector(0 to 35)	inout
enable	std_logic	in
ADC_CH1_value_untsg	unsigned(11 downto 0)	out

Die Komponente übernimmt die vom ADC über die GPIO seriell geschickten Daten, schreibt sie in einen unsigned Vector und leitet diesen über die Entity weiter. Es wird ausschließlich das MSB invertiert um im natürlichen Zahlenbereich zu bleiben. Das 10 MHz-Taktsignal wird intern über einen Clockteiler "ALTPPLL", aus dem QuartusPrime IP-Catalog, geteilt.

2.4 ADC-PCB "Prototype 1"

2.4.1 Projektteilbereich Übersicht

Zum Testen und erstmaligen Ansteuern des extrernen ADCs (siehe: 2.3) wurde ein extra PCB angefertigt. Der Schaltplan und das Platinenlayout wurden in KiCad gezeichnet, die Gerberdateien exportiert und die Platine anschließend mithilfe einer CNC-Fräsmaschiene 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.

2.4.2 Schaltplan

Der Schaltplan (Figure: 5) wurde möglist 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: ??).

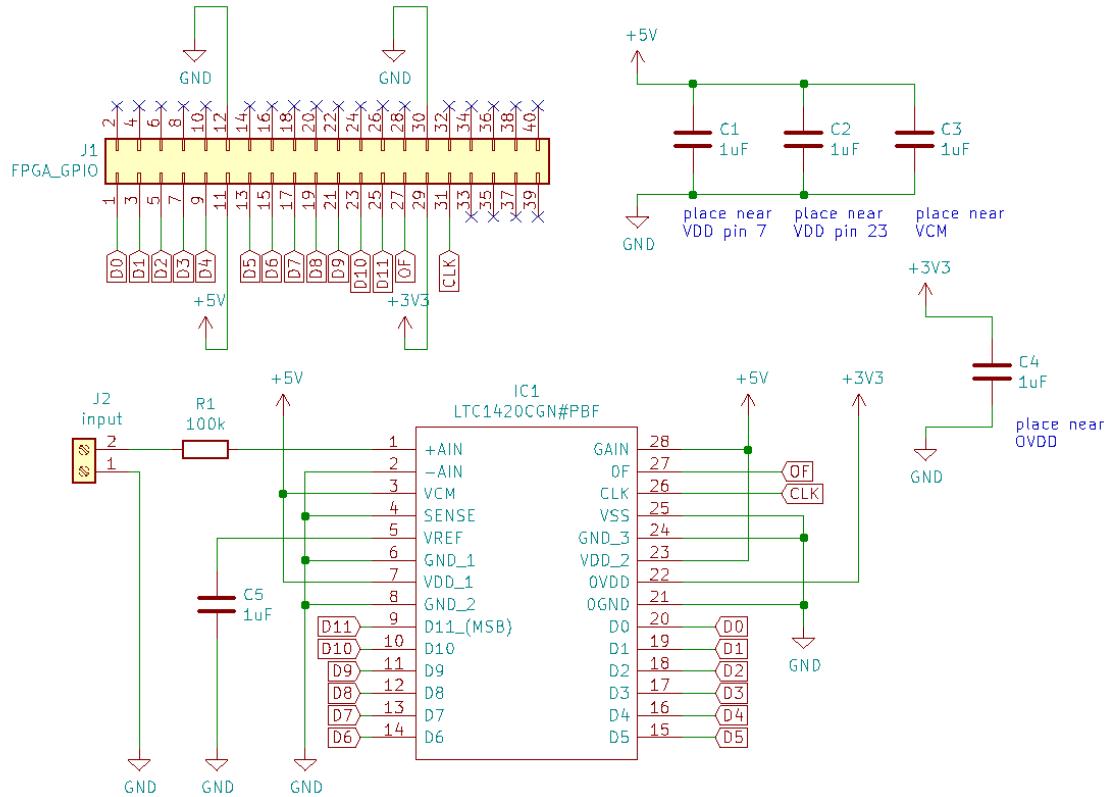


Figure 5: Schaltplan "Prototype 1"

2.4.3 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 8 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.

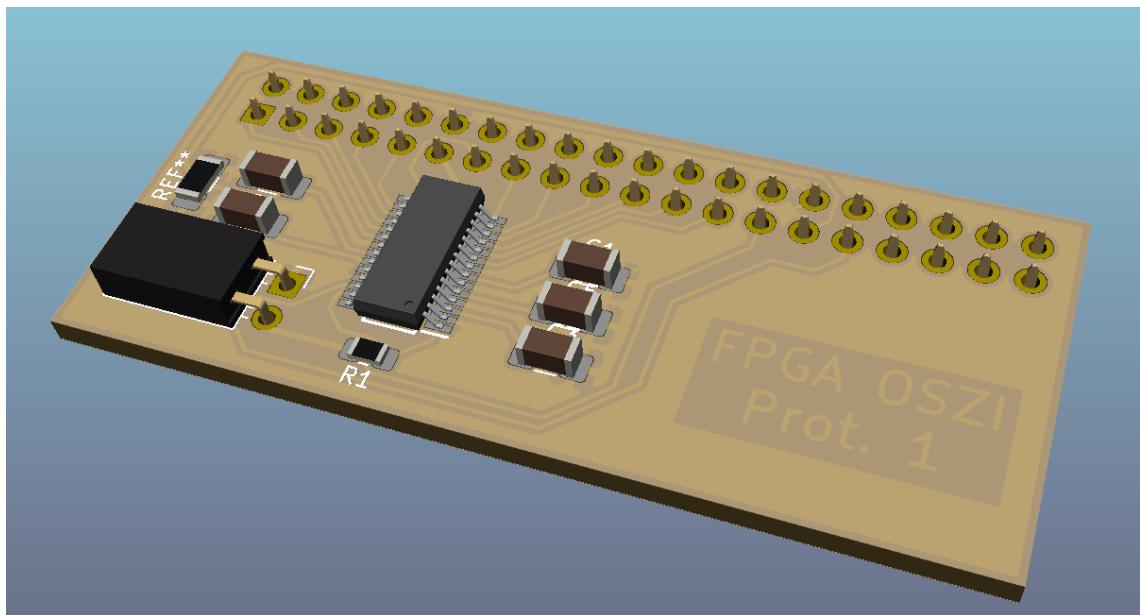


Figure 6: 3D-Ansicht Vorderseite

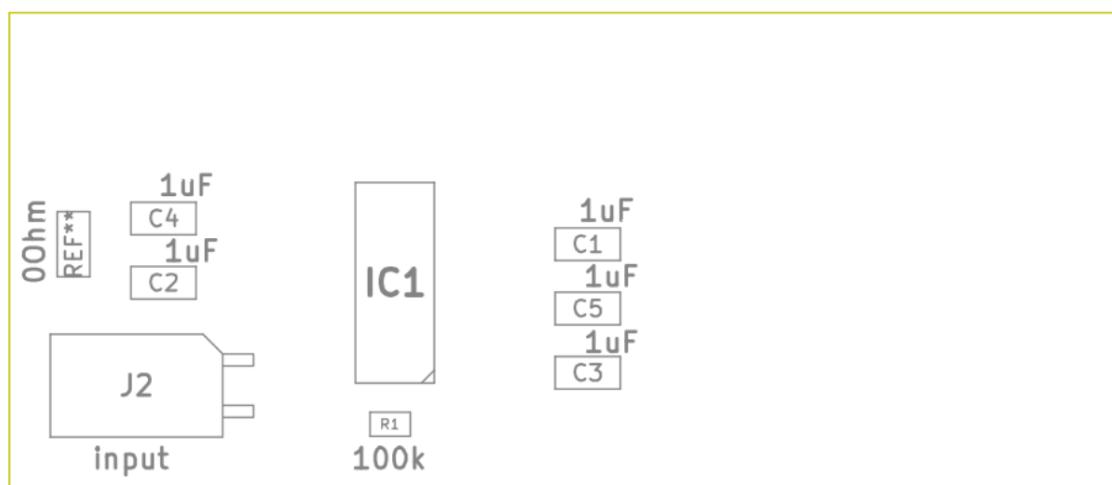


Figure 7: Bestückungsplan Vorderseite

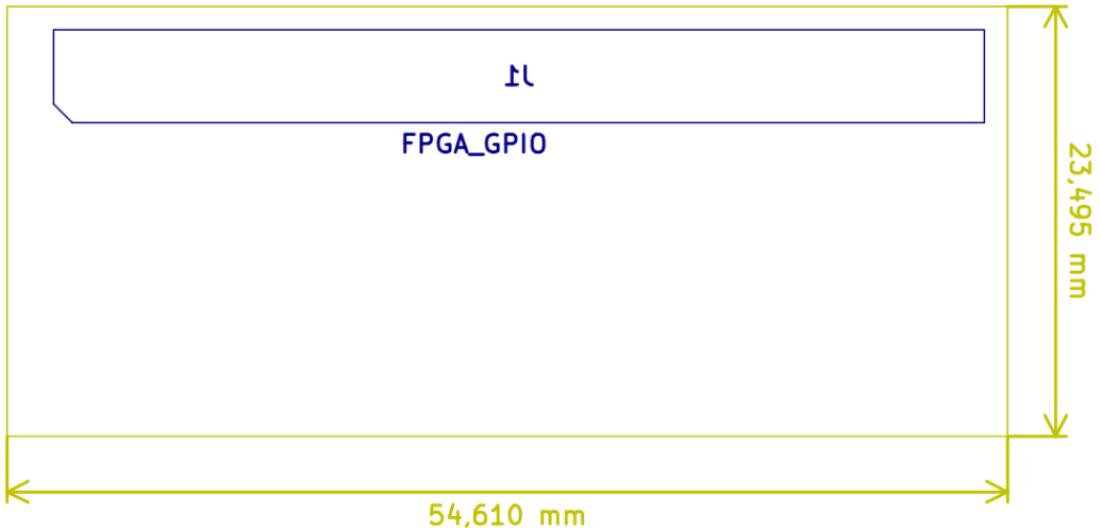


Figure 8: Bestückungsplan Rückseite & Bemaßung

2.5 Trigger

2.5.1 Projektteilbereich Übersicht

Die Komponente *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 Erreichen und 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) erreicht 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 9 sind 3 Szenarien Abgebildet, nur im ersten löst der Trigger aus.

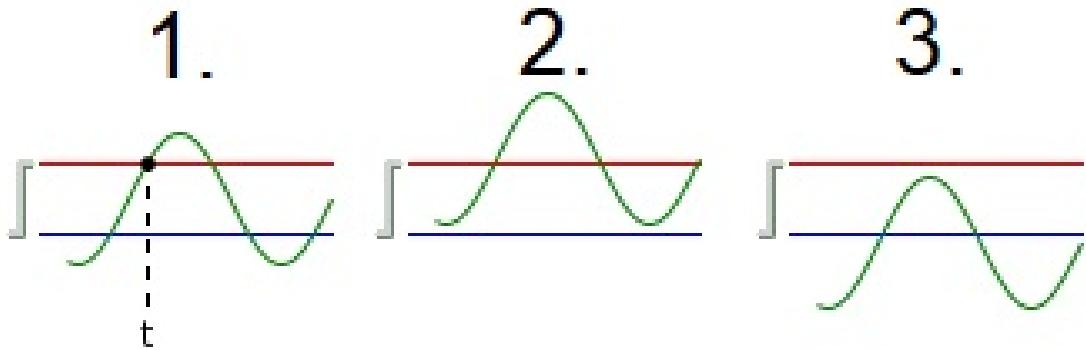


Figure 9: Trigger Rising-Edge-Detection

Modus: Falling Edge Bei der *Falling-Edge-Detection* löst der Trigger beim Erreichen oder 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**) erreicht werden (arming level_high). 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 10 sind 3 Szenarien Abgebildet, nur im ersten löst der Trigger aus.

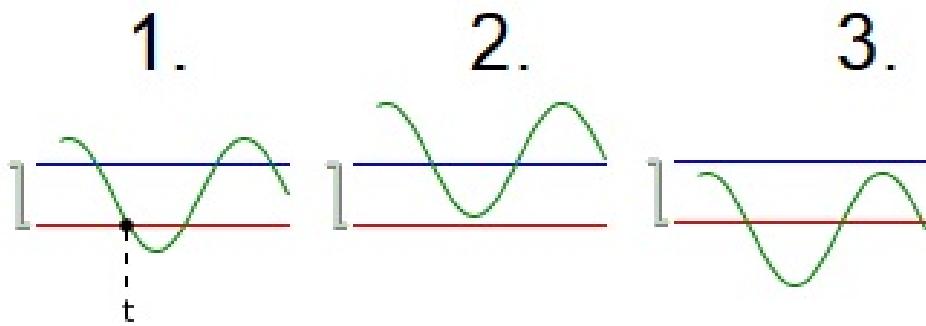


Figure 10: Trigger Falling-Edge-Detection

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 11 sind 4 Szenarien Abgebildet, in der ersten Zeile funktioniert das Triggern, in der zweiten nicht.

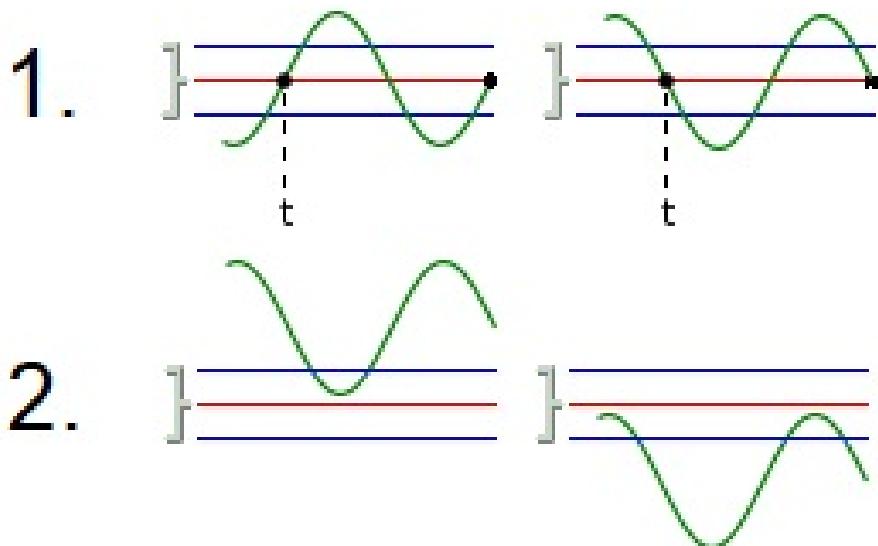


Figure 11: Trigger Any-Edge-Detection

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 50 digits wird empfohlen.

2.5.3 VHDL-Design-Unit

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

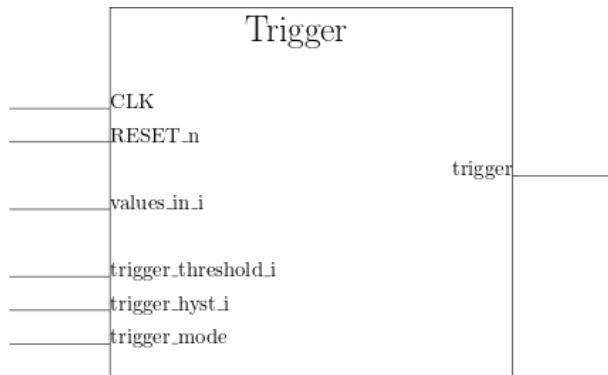


Figure 12: Entity Trigger

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 Simulation

Generelles Um die Komponente *Trigger* zu testen wurde eine Testbench geschrieben und mithilfe dieser eine Simulation in ModelSim durchgeführt. Neben der Clock, dem Reset und den Konfigurationseinstellungen wurde als zu triggernder Datenstream ein Dreieckssignal angelegt.

Die Design-Unit wurde gut durchgeprüft. Viele Testversuche mit möglichst verschiedenen und auch exotischen Konfigurationseinstellungen ergaben eine fehlerlose Funktionalität. Das Synthesisieren der Komponente stellt ebenfalls kein Problem dar.

Simulation Im abgebildeten Zeitdiagramm (Figure 13) ist eine der Simulationen zu sehen. Die Komponente wurde auf den Trigger-Mode *Any* gestellt,

was bedeutet, dass sie auf beide Flanken reagiert. Eine Hysterese von 20 digits und ein Schwellwert (=threshold) von 2100 digits wurden eingestellt. Es ist zu erkennen, dass die einzelnen, internen Signale aus der Komponente *trigger_falling* und *trigger_rising* zum richtigen Zeitpunkt ausgelöst werden. Diese ergeben im Trigger-Mode *any* oder-verknüpft das **rot** dargestellte Signal *trigger*.

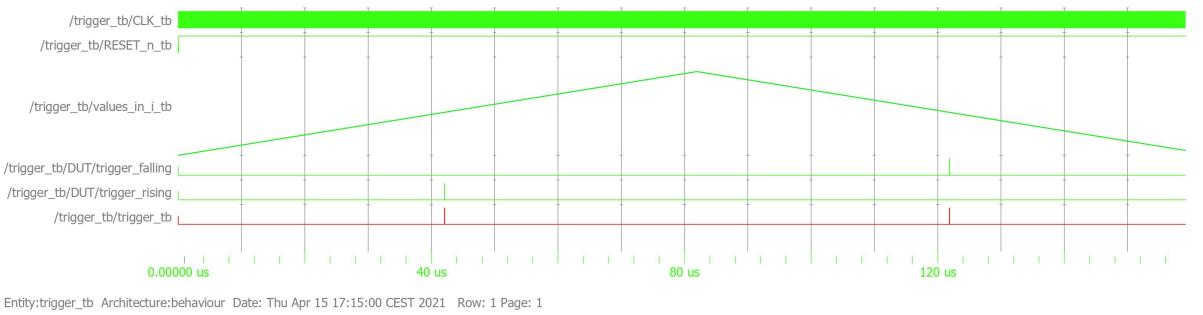


Figure 13: Simulation

2.5.5 Trigger Praxistest

Zusätzlich zur Simulation wurde der Trigger mit einem Messaufbau (14) getestet.

Messaufbau An den externen ADC auf der Platine *Prototyp 1* (siehe: 2.4) wurden verschiedene Signale angelegt. Hier zu sehen ist ein 500 kHz Rechtecksignal mit einer Amplitude von ca. 4 Vpp, das auf beide Flanken getriggert wird. Die Trigger-Leitung im FPGA ist über einen GPIO (GPIO(35)) herausgeführt und wird ebenfalls am Oszilloskop aufgezeichnet. Es wurden alle 3 Triggermodi durchgetestet.

Gerät	Referenz
Oszilloskop 1	3.2
Funktionsgenerator 1	3.3
Prototyp 1	2.4
FPGA	3.1

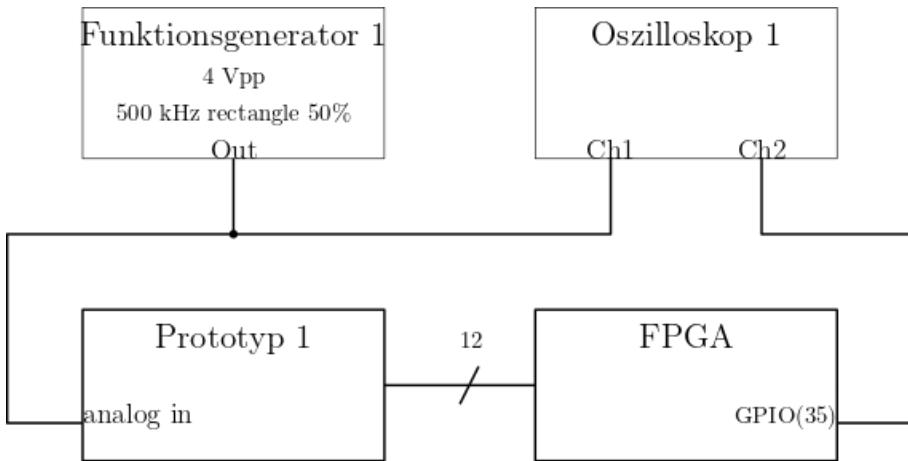


Figure 14: Messaufbau Triggertest

Messerkenntnisse Der Praxistest ergab, dass die Trigger-Unit korrekt funktioniert, obwohl dies auf den ersten Blick nicht danach aussieht. Das Bild 15 zeigt das Messergebnis am Bildschirm des Oszilloskopes, unter den in "Messaufbau" spezifizierten Umständen. Es ist zu sehen, dass sowohl der Trigger für die *rising edge*, als auch für die *falling edge* um etwa 390 ns zu spät am Bildschirm erscheint.

Die Erklärung hierzu ist einfach, wenn man sich die Timing-Diagramme ansieht. Wie in der Grafik 16 aus dem Datenblatt zu sehen, benötigt der ADC ca. $t_{delay} = 270\text{ns}$. Hier hinzu kommen noch zwei Taktzyklen des FPGAs, was 40ns gesamt sind, für den internen Auslösemechanismus. Die restlichen 60ns sind auf Signallaufzeiten und Messfehler zurückzuführen.

$$t_{conv} = 70\text{ns}$$

$$t_{clock} = \frac{1}{10\text{MHz}} = 100\text{ns}$$

$$tdelay = t_{conv} + 2 \cdot t_{clock} = 70 + 2 \cdot 100 = 270\text{ns}$$

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



Figure 15: Trigger-Test Oszilloskop1

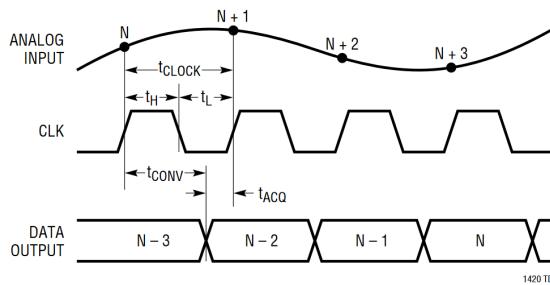


Figure 16: Timing-Diagram ADC

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

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

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

words insgesamt: 1214 bits insgesamt: 1608

Figure 17: Paketspezifikation FPGA zu UserInterface

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

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

words insgesamt: 22 bits insgesamt: 264

Figure 18: Paketspezifikation UserInterface zu FPGA

Wert / Nummer	Messbereich
0	1 : 1 (beispiel)
1	1 : 10 (beispiel)
2	1 : 24 (beispiel)

die durchnummerierten Messbereiche zu finden.

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 setzen 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.

2.7 UART - Verbindung FPGA & UserInterface

2.7.1 Projektteilbereich Übersicht

In diesem Abschnitt wird die Verbindung zwischen FPGA und UserInterface dokumentiert. Das Protokoll wird in einem eigenen Punkt erläutert (siehe: 2.6))

2.7.2 erste Verbindung

Die erste Verbindung wurde nicht über den am Board integrierten Kommunikations-Chip hergestellt, sondern über einen handelsüblichen RS232-Adapter "AU002E". Dabei wurde die Hardware und die UART-Design-Unit getestet und erprobt.

Pegelanpassung Der verwendete Adapter ist für RS232 ausgelegt, was der Vorläufer von UART ist, und hat daher ungünstige Eigenschaften. Die Pegel sind folgend definiert:

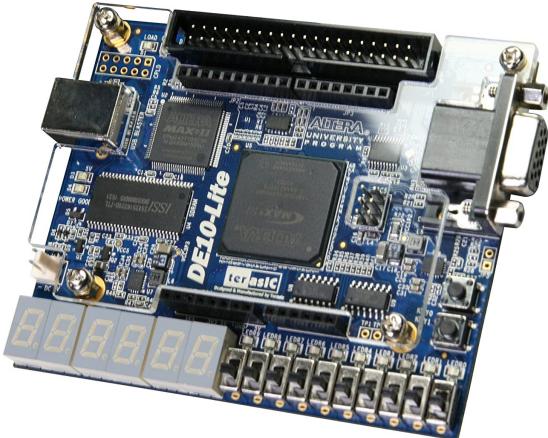
- logisch '1': zwischen -15V und -3V
- logisch '0': zwischen 3V und 15V

Überraschenderweise funktionierte das Senden der Daten in die Richtung vom FPGA zum Adapter, obwohl die Pegel '1' = 0V und '0' = 3.3V nicht den Spezifikationen entsprachen. Der Chip kann die digitalen Ausgänge ausschließlich auf diese zwei Spannungen schalten.

Hingegen die andere Richtung stellte ein Problem dar: An den Eingangspins des DE10-Lite boards dürfen maximal 3,3 V angelegt werden.

3 verwendete Messgeräte & Entwicklungstools

3.1 DE10-Lite Board

Art	DE-Lite Board
Produktnummer	DE10-Lite
Seriennummer	19040005-1757
TGM Inv. Nr.	Eigentum Sauer
Beschreibung	FPGA-Entwicklungsplatine Chip: Max 10 10M50DAF484C7G 50 MHz Power Supply: 5V USB
Foto	

3.2 Oszilloskop 1

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

3.3 Funktionsgenerator 1

Art	Funktionsgenerator
Marke	HP
Produktnummer	H3312A
Seriennummer	1502G00232
TGM Inv. Nr.	Eigentum Sauer
Beschreibung	analoger Funktionsgenerator 1 Channel 13 MHz sinus/rechteck/dreieck
Foto	

3.4 RS232-Adapter

Art	RS232-Adapter
Produktnummer	AU0002E
Seriennummer	/
TGM Inv. Nr.	Eigentum Sauer
Beschreibung	Umsetzter RS232 i-z USB DSUB 9 Pin männlich USB-A männlich
Foto	

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>
- UART VHDL Code: <https://www.digikey.com/eewiki/pages/viewpage.action?pageId=59507062>

Datenblätter:

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