

# digitales Theremin

## Fachbericht

PROJEKT 5  
10. August 2020

**Auftraggeber:** Prof. Dr. Hanspeter Schmid

**Betreuung:** Prof. Dr. Hanspeter Schmid  
Herr Prof. Karl Schenk

**Team:** Andreas Frei  
Dennis Aeschbacher

**Studiengang:** Elektro- und Informationstechnik

**Semester:** Herbstsemester 2019

## **Abstract**

In this Project a Theremin was built that mainly operates on digital hardware unlike the original device that solely used analog electronics. The device is supposed to be used in presentations for trade fairs by the Institute for Sensors and Electronics ISE. As such the device should be built in a appealing housing. Moreover the device should have other additional functionality such as soundeffects or the ability to record sound. The digital hardware was implemented in VHDL on the developer board DE1-SoC from terasIC with a Cyclone V FPGA from Intel. The sole analog component implemented was the oscillator that controls the pitch. The pitch of the device can be changed well, but the sound itself has a flaw at the moment, because there is an audible crackle. This is due to a communication problem with the codec that was used for the audio output. This problem will not be corrected during this project, because the communication will be implemented differently in the finished product. The work in this project served as a platform for the continuation in project 6. The next steps would be to implement the volume control and redesign the pcb for two antennas and oscillators.

## Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
<b>2 Technische Grundlagen</b>	<b>2</b>
2.1 analoges Theremin . . . . .	2
2.2 digitales Theremin . . . . .	3
<b>3 Realisierung</b>	<b>7</b>
3.1 Antennenoszillator . . . . .	7
3.2 Referenzoszillator . . . . .	8
3.3 Mischer . . . . .	9
3.4 Filter . . . . .	9
3.5 Tongenerierung . . . . .	10
<b>4 Validierung</b>	<b>12</b>
4.1 Inbetriebnahme PCB . . . . .	12
4.2 Simulation . . . . .	12
4.3 Debugging . . . . .	16
<b>5 Schlusswort</b>	<b>18</b>
<b>6 Ehrlichkeitserklärung</b>	<b>19</b>
<b>Literatur</b>	<b>20</b>
<b>A Projektklärung</b>	<b>21</b>
<b>B Schema Antennenoszillator</b>	<b>31</b>
<b>C PCB Antennenoszillator</b>	<b>32</b>

## 1 Einleitung

Das Theremin kennen heutzutage nur wenige Leute, obwohl es das erste elektronische Instrument war. Es wurde 1920 von dem Russen Lev Sergejewitsch Termen, welcher sich später zu Leon Theremin umbenennen liess, erfunden [1]. Personen die regelmässig Filme schauen, haben die Musik welche mit einem Theremin gemacht wird bestimmt schon einmal gehört. Ein Beispiel dafür ist Ghostbusters, wo das Theremin oft im Hintergrund zu hören ist. Zudem ist das Theremin in einigen Science-Fiction-Filmen und Horrorfilmen zu hören [2]. Das Theremin wird ohne es zu berühren gespielt, indem man mit den Händen die Distanz zu zwei Antennen ändert. Dies führt zu Veränderung der Tonhöhe und Lautstärke.

Im Projekt 5 und 6 soll nun ein solches Instrument entwickelt werden. Mit dem Unterschied, dass das sonst analoge Instrument digital aufgebaut werden soll. Dabei soll es auf einem Field Programmable Gate Array (FPGA) implementiert werden. Später soll das Theremin als Messobjekt für das Institut für Sensorik und Elektronik ISE verwendet werden. Im Rahmen des Projekt 5 wurde die Tonhöhenantenne des Theremin realisiert. Dazu wurde die Antenne zusammen mit dem Antennenoszillator analog beibehalten. Die restlichen Komponenten wurden in VHDL realisiert. Das Resultat wurde auf dem DE1-SoC Board von terasIC mit einem Cyclone V FPGA von Intel getestet.

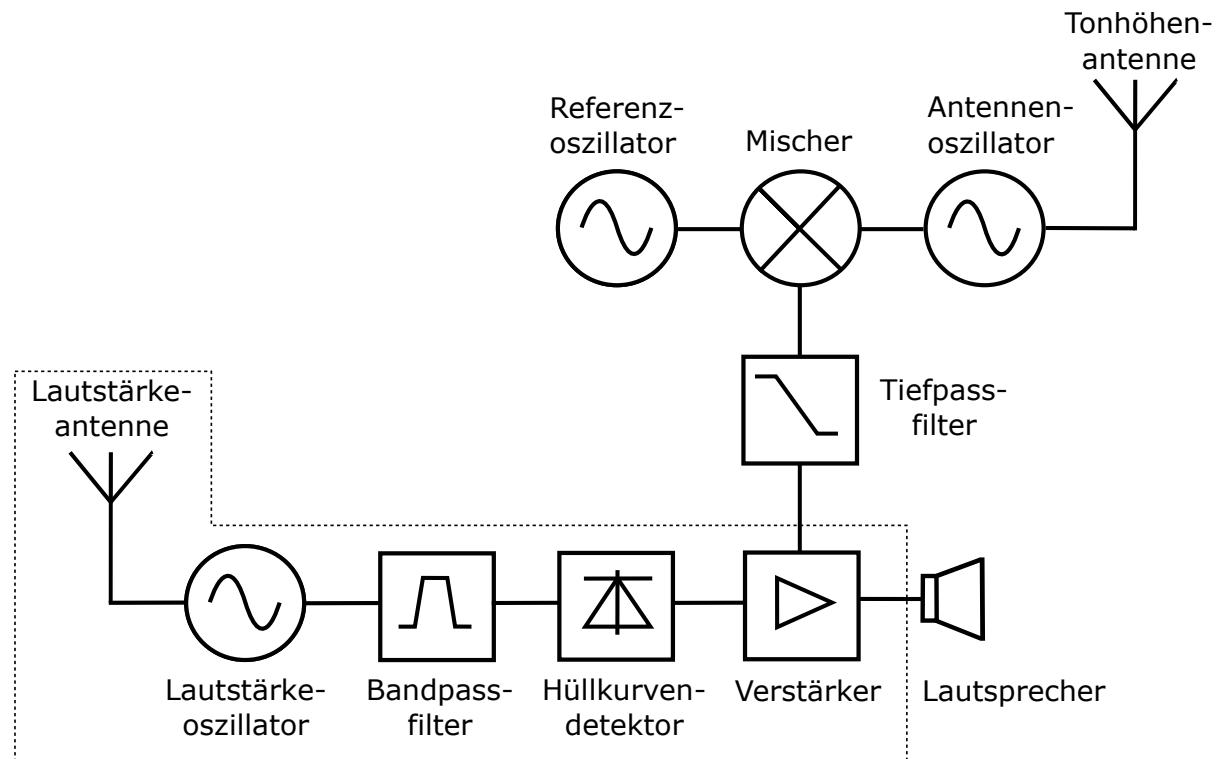
Der folgende Fachbericht beginnt mit dem Kapitel 2 Technische Grundlagen. In der ersten Hälfte des Kapitel wird erklärt wie ein analoges Theremin funktioniert und welche Komponenten ein Theremin ausmachen. In der zweiten Hälfte werden digitale Lösungsansätze besprochen. Anschliessend wird im Kapitel 3 Realisierung beschrieben wie die Komponenten realisiert wurden. Im Kapitel 4 Validierung wird als erstes auf die Inbetriebnahme des Antennenoszillators eingegangen. Als nächstes werden die Simulationen des VHDL Codes erläutert. Im letzten Abschnitt wird auf die Inbetriebnahme des VHDL Codes auf dem DE1-SoC Board Bezug genommen.

## 2 Technische Grundlagen

In diesem Kapitel wird erklärt wie ein Theremin analog funktioniert. In der zweiten Hälfte des Kapitels wird das Theremin mit digitalen Komponenten vorgestellt.

### 2.1 analoges Theremin

Das klassische Theremin besitzt zwei Antennen. Über die senkrecht angebrachte Antenne kann die Tonhöhe beeinflusst werden. Die zweite Antenne ist waagrecht angeordnet und beeinflusst die Lautstärke. Im Projekt 5 wird die Antenne für die Lautstärke Regelung nicht realisiert. Daher wird in den Technischen Grundlagen auch nicht weiter darauf eingegangen. Die Tonhöhe des Theremin kann über die Handposition zur Antenne beeinflusst werden. Dadurch wird das von einem Antennenoszillator erzeugte elektrische Feld beeinflusst. Die Frequenz dieser Oszillatoren ist jedoch weit über dem hörbaren Bereich (zwischen 100 kHz bis 1 MHz). Mit Hilfe eines Mixers und einem Referenzoszillator wird die Frequenzdifferenz hörbar gemacht und danach verstärkt. Die einzelnen Schaltungsteile werden nun noch genauer erklärt, dabei wird auf das Bauset „Theremin selber bauen“ von Franzis Bezug genommen [3].



**Abbildung 2.1:** Blockschaltbild eines analogen Theremins. Die Komponenten innerhalb der gestrichelten Linie werden im P6 realisiert

#### Antennenoszillator und Tonhöhenantenne

Die Tonhöhenantenne ist ein Metallrohr welches mit dem Antennenoszillator verbunden ist. Durch die Hand des Spielers wird über die Tonhöhenantenne die Frequenz des Antennenoszillators verändert. Die Kapazitätsänderung welche über die Antenne erreicht werden kann ist sehr gering. Diese liegt im Picofarad Bereich [4]. Um eine genug grosse Frequenzänderung zu

erzeugen muss demnach die Grundfrequenz des Antennenoszillators weit über dem hörbaren Frequenzbereich gewählt werden.

Im Franzis Bauset wurde diese auf ca. 500 kHz gesetzt. Der Antennenoszillator wurde mit einer Colpitts-Schaltung realisiert [3].

### Mischer und Referenzoszillator

Die erzeugte Frequenz der Tonhöhenantenne ist weit über dem vom Menschen hörbaren Bereich. Deswegen wird das Antennen Signal mit einem Referenzoszillator mit fester Frequenz gemischt. Dies bewirkt eine Frequenzdifferenz im hörbaren Bereich. Um diese Differenz zu erhalten multipliziert der Mischer die zwei Signale des Referenzoszillators  $A_1 \sin(\omega_1)$  und des Antennenoszillators  $A_2 \sin(\omega_2)$  wie folgt:

$$V_{out} = A_1 A_2 \sin(\omega_1 t) \sin(\omega_2 t) \quad (2.1)$$

$V_{out}$  kann durch Additionstheoreme umgeformt werden. Dabei erhält man folgenden Ausdruck:

$$V_{out} = A/2[\cos((\omega_1 - \omega_2)t) - \cos((\omega_1 + \omega_2)t)] \quad (2.2)$$

Das Ausgangssignal  $V_{out}$  hat zwei Frequenzkomponenten. Zum einen die Differenz der beiden Frequenzen zum anderen die Summe der Frequenzen. Dabei ist bei dem Theremin nur die Differenz der Frequenzen von Interesse [4].

Das Theremin muss vor jedem Gebrauch kalibriert werden. Es könnte beispielsweise sein, dass die Differenz der Frequenz ausserhalb des hörbaren Bereiches ist. Dazu wird mit Hilfe eines Trimmkondensators am Referenzoszillator die Differenzfrequenz auf 0Hz gestellt.

Das Franzis Bauset realisiert den Mischer mit zwei bipolaren Transistoren. Der Referenzoszillator wurde mit derselben Colpitts-Schaltung wie zuvor der Antennenoszillator realisiert. Dies mit dem Unterschied des oben genannten Trimmkondensators für die Kalibrierung [3].

### Tiefpassfilter

Mit Hilfe eines Tiefpassfilters wird das Signal mit der Summe der Oszillator Frequenzen vollständig abgeschwächt. Übrig bleibt die Differenz der Oszillator Frequenzen. Dieser ist der Anteil des Mischprozesses, welcher von Interesse ist, da er im hörbaren Bereich ist.

$$V_{out} = A/2\cos((\omega_1 - \omega_2)t) \quad (2.3)$$

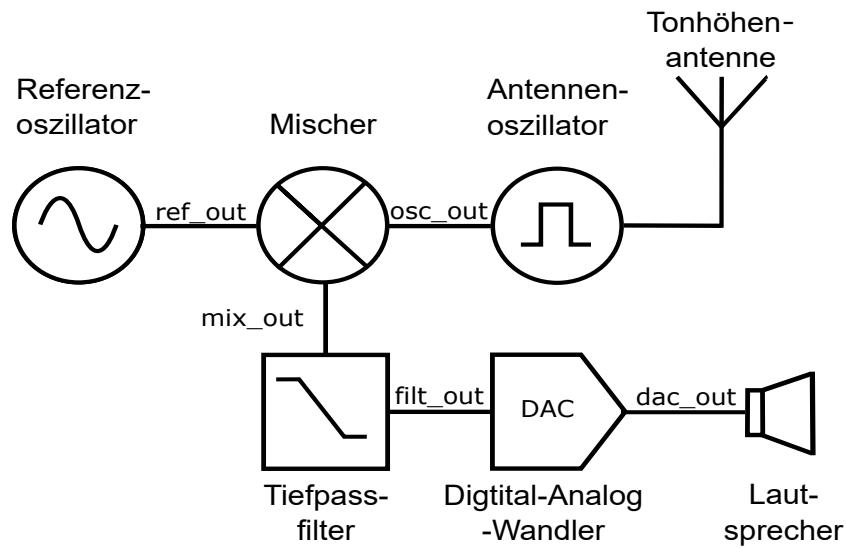
Im Franzis Bauset ist der Tiefpass mit einem RC Glied realisiert [3].

### Verstärker und Lautsprecher

Die hörbare Differenz wird verstärkt und über einen Lautsprecher ausgegeben. Das Franzis Bauset realisiert die Verstärkung über ein Audioverstärker IC [3].

## 2.2 digitales Theremin

Wie schon am Anfang dieses Kapitels angedeutet, werden in diesem zweiten Teil Möglichkeiten für den Aufbau des Theremins als ein digitales Instrument aufgezeigt.



**Abbildung 2.2:** Blockschaltbild digitales Theremin mit eingezeichneten Ausgängen.

### Antennenoszillator

Um den Oszillatot zur Änderung der Tonhöhe und der Lautstärke zu realisieren waren mehrere Optionen möglich. Diese wurden in der Projektklärung genauer verglichen (siehe Anhang). Schlussendlich wurde entschieden, dass dieser Teil des Theremins weiterhin analog bleiben soll, um das Theremin wie anhin spielen zu können, ohne einen extremen Aufwand treiben zu müssen. Das generierte Signal wird an *osc\_out* ausgegeben.

### Referenzoszillator

Der zweite Oszillatot wird nun digital realisiert. Dazu wurden zwei Möglichkeiten in Erwägung gezogen. Zum einen ein Oszillatot mithilfe von Lookup Tables. Diese brauchen keine komplizierte Berechnung sondern nur eine Tabelle mit den Sinuswerten einer viertel Periode. Diese würden dann so ausgegeben, dass eine ganze Periode entsteht. Die zweite und etwas aufwendigere Variante ist ein Sinusgenerator mithilfe des Cordic Algorithmus. Bei diesem werden die Werte erst während der Laufzeit berechnet. Dazu wird zum einen eine Komponente benötigt, welche den Sinus Wert berechnet und zum anderen eine Komponente welche die entsprechenden Winkelwerte in der richtigen Frequenz bereitstellt.

Es wurde entschieden den Cordic Algorithmus einzusetzen, da dieser zeitgemässer ist und einen grösseren Lerngewinn bietet.

Der Cordic Algorithmus wird gebraucht, um verschiedenste mathematische Operationen iterativ grösstenteils nur mit Additionen und Verschiebung von Bits zu berechnen. Im Falle des Theremins wurde dieser gebraucht, um aus einem vorgegebenen Winkel den Sinus zu berechnen. Der Cordic Algorithmus kann in zwei Modi betrieben werden. Zum einen der Vektor Modus, mit welchem von einem gegebenen Vektor der Winkel berechnen werden kann und zum anderen der Rotationsmodus, mit welchem aus einem gegebenen Winkel die Elemente des zugehörigen Vektors berechnet werden können. Für die beiden Modi werden folgende Formeln verwendet [5]:

$$x_{i+1} = x_i - y_i d_i 2^{-i} \quad (2.4)$$

$$y_{i+1} = y_i + x_i d_i 2^{-i} \quad (2.5)$$

$$z_{i+1} = z_i - d_i \arctan 2^{-i} \quad (2.6)$$

Dabei wird  $d_i$  im Rotationsmodus wie folgt berechnet:

$$d_i = \begin{cases} -1 & z_i < 0 \\ 1 & \text{otherwise} \end{cases} \quad (2.7)$$

Wie man sieht werden in jeder Iteration die neuen Werte aus den alten Werten berechnet. Um den Algorithmus so einzusetzen, dass aus einem gegebenen Winkel der Sinus und der Kosinus berechnet wird, müssen die benötigten Anfangsbedingungen wie folgt gewählt werden:

$$\begin{aligned} x_0 &= 1 \\ y_0 &= 0 \\ z_0 &= \varphi \end{aligned} \quad (2.8)$$

$\varphi$  ist der gegebene Winkel, welcher zwischen  $-\pi/2$  und  $\pi/2$  sein muss, damit der Algorithmus konvergiert.

Daraus ergeben sich nach  $n$  Iterationen der Sinus und Kosinus Wert wie folgt:

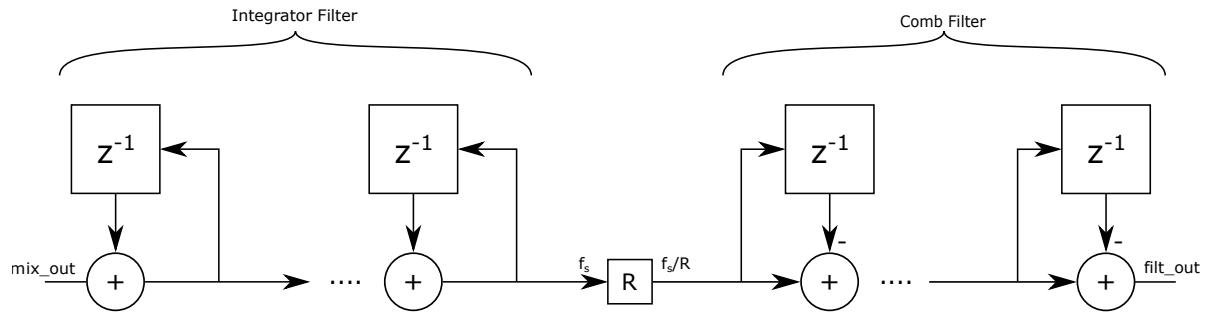
$$\begin{aligned} x_n &= \frac{\cos \varphi}{A} \\ y_n &= \frac{\sin \varphi}{A} \end{aligned} \quad (2.9)$$

Somit müssen am Schluss noch die erhaltenen Werte mit der Konstante  $A = 0.60725294$  multipliziert werden um die genauen Sinus- und Kosinuswerte zu erhalten. Diese Resultate werden am Ausgang *ref\_out* ausgegeben.

### Mischer

Der Mischer kann als digitale Komponente sehr einfach realisiert werden. Er muss lediglich die beiden Signale der Oszillatoren *osc\_out* und *ref\_out* miteinander multiplizieren. Weiter wurde das Signal *osc\_out* in ein Rechtecksignal gewandelt. Dies ermöglicht es auf einen Analog-Digital-Wandler zu verzichten. Dies ändert nichts an der Funktionalität des Theremins, da die Oberwellen des Rechtecksignals eine so hohe Frequenz haben, dass sie nie in die Nähe der hörbaren Frequenzen kommen. Auch nach dem mischen haben die Produkte zwischen Referenzoszillator und Oberwellen noch so hohe Frequenzen, dass dies keine Probleme bereitet. Das gemischte Resultat wird am Ausgang *mix\_out* ausgegeben.

## Tiefpassfilter



**Abbildung 2.3:** Aufbau eines CIC-Filters N-ter Ordnung mit Eingang *mix\_out* und Ausgang *filt\_out*

Um das gemischte Signal am Ende mit einem Tiefpass zu filtern, sind verschiedene Methoden möglich. Zum einen können FIR oder IIR Filter eingesetzt werden. Da das Signal am Ende unterabgetastet werden muss, um die richtige Frequenz für den Digital-Analog-Wandler zu erhalten, war die Verwendung eines CIC-Filters ideal. Dieses hat zudem den Vorteil, dass keine Multiplikationen durchgeführt werden müssen, da es lediglich Additionen und Subtraktionen benötigt [6].

Wie man in Abbildung 2.3 sieht ist das Filter in drei Stufen aufgeteilt. Die erste Stufe bildet der Integratorfilter, welcher alle Eingangswerte aus *mix\_out* aufsummiert. Dabei ist der entstehende wrap-around durchaus gewünscht. In der zweiten Stufe wird die Abtastrate  $f_s$  des Signals um den Faktor  $R$  dezimiert. Die dritte Stufe enthält den Combfilter, welcher mit einer tieferen Abtastrate die Daten verarbeitet. Dabei wird der aktuelle Abtastwert mit dem letzten subtrahiert und an *filt\_out* ausgegeben. Zudem können mehrere Integrator- und Combfilterpaare hintereinandergeschalten werden um die Ordnung des Filters zu erhöhen. Dieses Filter hat im Vergleich zur normalen Unterabtastung weniger Aliasing zur Folge [7].

## Digital-Analog-Wandler

Um die diskreten Daten als Ton auszugeben wird ein Digital-Analog-Wandler (DAC) benötigt. Damit der DAC korrekt funktioniert, müssen die Daten an *filt\_out* mit einer bestimmten Abtastfrequenz zur Verfügung gestellt werden. Das analoge Signal wird an *dac\_out* an den Lautsprecher ausgegeben.

### 3 Realisierung

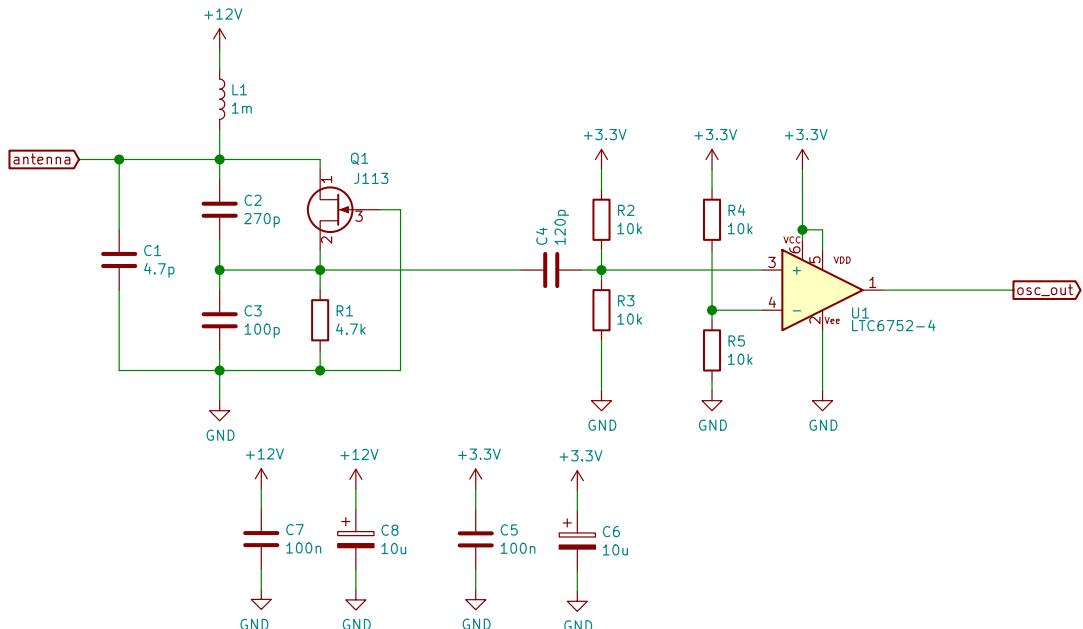
Um die digitalen Komponenten zu realisieren wurde das FPGA Entwicklungsboard DE1-SoC von terasic mit einem Cyclone V FPGA von Intel zur Verfügung gestellt. Dieses hat diverse Peripherien welche für die Realisierung nützlich sind. Weiter wurde für die Einsynchronisation der Taster die Intellectual Property (IP) aus dem Fach digitale Schaltungstechnik (dst) verwendet.

#### 3.1 Antennenoszillator

Der Antennenoszillator wurde analog auf einem PCB realisiert. Dabei wurde die gleiche Oszillator Schaltung verwendet wie im Franzis Bauset. Die Schaltung ist eine Colpitts-Oszillator Schaltung mit einem JFET. Damit das Sinussignal des Antennenoszillators nicht A/D gewandelt werden muss, wurde entschieden das Sinussignal in ein Rechtecksignal mit gleicher Frequenz zu wandeln. Dies geschieht mithilfe einer Komparatorschaltung.

Wie in Abbildung 3.1 zu sehen ist wird das PCB mit 12 V und 3.3 V DC betrieben. Diese Spannungen werden von Labornetzgeräten erzeugt. Um die Versorgungsspannung zu stabilisieren wurden für beide Spannungen ein 100 nF Keramikkondensator und ein 10 uF Elektrolytkondensator verbaut. Der Colpitts-Oszillator wird wie im Franzis Bauset mit 12 V DC betrieben. Dies ist nötig um ein genügend starkes elektrisches Feld der Antenne zu erzeugen. Im Versuchsaufbau wurde versucht den Colpitts-Oszillator mit 3.3 V zu betreiben. Mit 3.3 V ist jedoch das elektrische Feld der Antenne zu klein. Der Spieler kann nur sehr nahe an der Antennen das elektrische Feld beeinflussen. Die Komparatorschaltung wird mit 3.3 V betrieben da die Logikeingänge des FPGA auf diese Spannung ausgelegt sind, und das Signal *osc\_out* diesen Pegel haben muss.

Als Antenne wird ein Messingrohr verwendet. Dieses ist zwischen der Spule L1 und dem Kondensator C1 verbunden. Da der im Bauset verwendete JFET nicht mehr bestellbar ist, wurde der J113 N-Channel JFET verwendet. Die mit LTspice simulierten Werte des J113 glichen stark der original Schaltung, weshalb dieser gewählt wurde.



**Abbildung 3.1:** Schema Antennenoszillator. Links Collpitts-Oszillator, rechts Komparatorschaltung

Die Ausgangsspannung des Colpitts-Oszillators wird über den Kondensator C4 entkoppelt. Dabei wird der DC-Anteil entfernt. Der Kondensator C4 und die Widerstände R2 und R3 bilden zusammen einen Hochpass. Damit die Oszillatoren Frequenz von ca. 570 kHz das Filter passieren kann wurde C4 so gewählt, dass die Grenzfrequenz des Filters bei ca. 265 kHz liegt.

Um das Sinussignal des Colpitts-Oszillators in ein Rechtecksignal zu wandeln, wurde eine nichtinvertierende Komparatorschaltung gewählt. Dazu wurde der LTC6752 Komparator mit Rail to Rail und Single Supply verwendet [8]. Als Referenzspannung wurde die halbe Versorgungsspannung gewählt. Diese wurde mit den Widerständen R4 und R5 realisiert.

### 3.2 Referenzoszillator

Wie schon in Kapitel 2.2 erwähnt wurde der Referenzoszillator mithilfe des Cordic Algorithmus realisiert. Dieser ist ideal für Anwendungen in einem FPGA, da er grösstenteils mit Additionen und Bit Verschiebungen auskommt und nur eine Multiplikation am Schluss durchführen muss. Um den Algorithmus in einem Sinusgenerator einsetzen zu können, wurde der Generator in zwei Komponenten aufgeteilt. Zum einen der Controller, welcher den Momentanen Winkel vorgibt und zum anderen der Prozessor, welcher den entsprechenden Sinuswert berechnet und an *ref\_out* ausgibt.

Da der Cordic Algorithmus nur für Winkelwerte zwischen  $-\pi/2$  und  $\pi/2$  konvergiert, wurden die Winkelwerte wie in Abbildung 3.2 dargestellt berechnet. Zuerst wird der Sägezahn Winkel berechnet. Dazu wird nach jedem Clockzyklus ein Zähler um einen Schrittwert hochgezählt. Dabei wird der wrap around des Zählers bewusst eingesetzt, um den Sprung zwischen II und III Quadranten zu erhalten. Der Schrittwert wird wie folgt berechnet:

$$step = \frac{2^{n+1} f_{sig}}{f_{clk}} \quad (3.1)$$

Wobei  $n$  die Anzahl Bits des Wertebereichs des Dreiecks Winkels ist,  $f_{sig}$  die gewünschte Frequenz des generierten Signals und  $f_{clk}$  die Clock Frequenz des FPGAs.

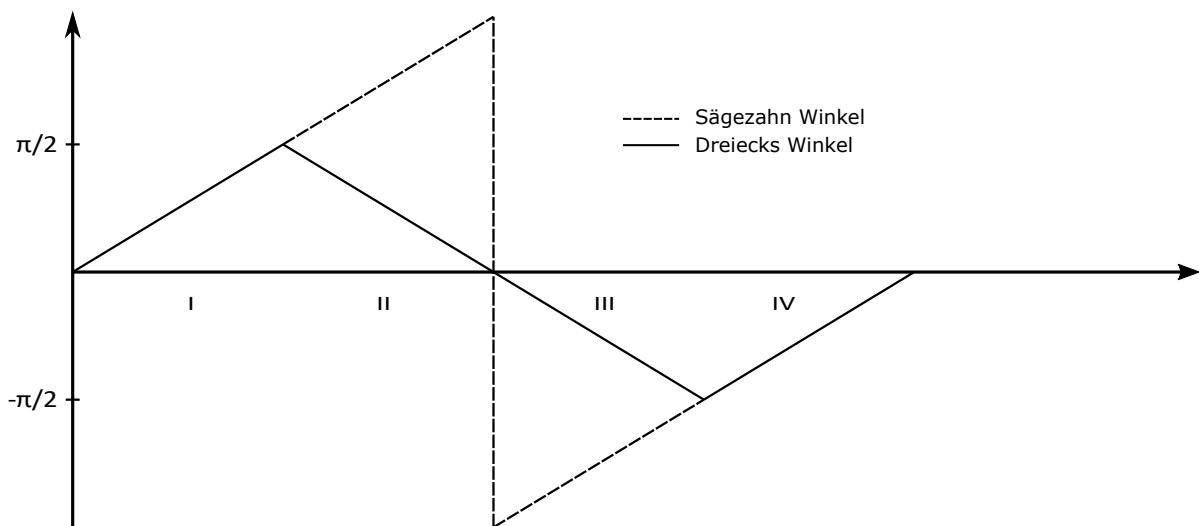


Abbildung 3.2: Winkel des Cordic Controllers

Um dann aus dem Sägezahn Winkel den Dreiecks Winkel zu generieren müssen die vordersten beiden Bits betrachtet werden. Sind diese entweder 01 oder 10 befindet sich das Signal im II respektive im III Quadranten. Ist dies der Fall müssen alle Bits ausser dem Vorzeichenbit (MSB) invertiert werden damit sich der Dreiecks Winkel ergibt. Somit werden, obwohl der Cordic nicht direkt die Werte im II und III Quadranten berechnen kann, alle vier Quadranten richtig berechnet [5].

Um nun den eigentlichen Cordic Algorithmus im FPGA zu implementieren müssen einige Dinge an der Theorie in Kapitel 2.2 angepasst werden.

Zum einen kann die Berechnung von  $\arctan 2^{-i}$  vorgängig gemacht werden und in eine Lookup Table gespeichert werden. Zum anderen wird der Algorithmus in einer Pipeline berechnet, da nach jedem Clockzyklus ein berechneter Wert anliegen soll. Und zuletzt wird nicht mit den Potenzen  $2^{-i}$  multipliziert sondern die entsprechenden Bits um  $i$  nach rechts verschoben.

Das Resultat wurde als eine signed Zahl definiert um die Berechnung von negativen Zahlen zu ermöglichen. Zudem wurden die Berechnungen im fixed-point Format durchgeführt, da die Berechnung von Nachkommazahlen nötig war. Ein Signalwert hat dabei 16 Bit mit einem Vorzeichenbit und 15 Nachkommastellen.

Der Referenzoszillator wird standardmäßig auf die Frequenz des Antennenoszillators eingestellt. Jedoch haben kleine Änderung der Umgebung schon Frequenzunterschiede zur Folge. Aus diesem Grund kann über zwei Taster in 100 Hz Schritten die Frequenz des Referenzoszillators verstellt werden um diesen auf den Antennenoszillator abzustimmen. Bis die Differenzfrequenz auf 0 Hz kalibriert ist.

### 3.3 Mischer

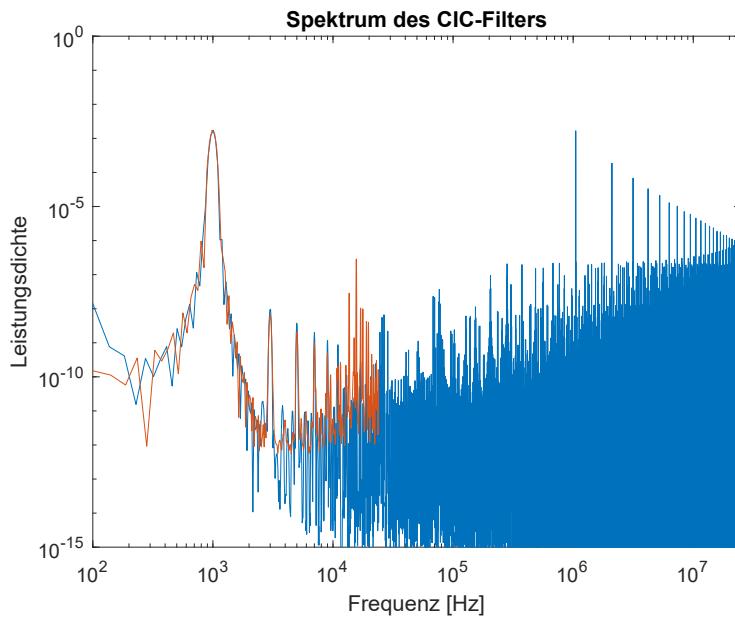
Der Mischer konnte, wie schon in Kapitel 2.2 erwähnt, sehr einfach in VHDL realisiert werden. Über ein GPIO wird das Rechtecksignal des Antennenoszillators *osc\_out* eingelesen. Das Rechtecksignal spiegelt die positive und negative Halbwelle des Antennenoszillators wieder. Dies signalisiert wann das Signal *ref\_out* mit +1 oder -1 multipliziert werden muss. Eine Multiplikation mit +1 ist unnötig und da eine Multiplikation mit -1 sehr trivial ist und zu viele unnötige Ressourcen verbraucht, wurde sobald das Rechtecksignal auf 0 ist das Zweierkomplement des Mischesignales gebildet. Diese Methode ermöglicht es den Mischer mit wenig FPGA Ressourcen zu realisieren.

### 3.4 Filter

Wie schon in Kapitel 2.2 erwähnt wurde das Tiefpassfilter mithilfe eines CIC-Filters realisiert. Jedoch musste um dieses implementieren zu können noch einige Größen bestimmt werden. Erstens muss für ein CIC-Filter die Bitanzahl des Ausgangssignals *filt\_out* im Vergleich zum Eingangssignal *mix\_out* noch erhöht werden. Dazu ist folgende Formel gegeben [7]:

$$B_+ = \lceil N \log_2 RM \rceil \quad (3.2)$$

M ist dabei die Verzögerung der Speicherelemente. N ist die Ordnung des Filters. R ist der Dezimierungsfaktor, um welcher die Abtastfrequenz verkleinert wird. Dieser ist hier 1000. M und N wurden vorerst beide als 1 angenommen. Somit ergab sich eine Erhöhung von 10 Bits im Vergleich zum Eingang um die Funktionalität gewährleisten zu können. Wie das Spektrum in Bild 3.3 zeigt, sind die angenommenen Parameter angemessen.



**Abbildung 3.3:** Berechnetes Spektrum des CIC-Filters mit Rechtecksignal (501 kHz) gemischt mit Sinussignal (500 kHz) unter Anwendung von Hanspeter Schmids Berechnungsmethode [13]

Um die Daten über den Avalon-Streaming Bus an die IP Komponente Audio Core übertragen zu können mussten hier beide generierten Clockfrequenzen verwendet werden. Dies, da der Audio Core die Clockfrequenz 12.288 MHz benötigt und für den Streaming Bus beide Seiten dieselbe Frequenz brauchen. Zudem musste hier auf den Clock-Domain Wechsel geachtet werden und die Steuerleitungen *valid* einsynchronisiert werden.

### 3.5 Tongenerierung

Zur Tongenerierung wird der auf dem DE1-SoC Board verbaute Audio Codec *WM8731* verwendet [9]. Dieser wandelt das digitale Signal in ein analoges um und gibt es am Line-Out Kopfhörer Ausgang aus. Für die Ansteuerung des Codec werden Intellectual Property Cores (IP-Cores) von Intel verwendet. Um mit dem Codec zu interagieren, werden drei IP Cores verwendet: der *Audio Core*, der *Audio and Video Config Core* und der *Audio clock Core*.

Der *Audio Core* wird verwendet um die digitalen Tondaten des CIC Filters über I<sup>2</sup>S an den Codec zu senden. Der Codec verlangt ein serielles 24 Bit langes Audio sample für den linken und den rechten Audio Kanal [10]. Die 32 Bit langen digitalen Tondaten des CIC Filters werden im Audio Core für den linken und rechten Tonkanal serialisiert. Die niederwertigsten 8 Bit der digitalen Tondaten werden abgeschnitten. Über das Sample Rate Left/Right Clock (DACLRC) Signal des Codecs wird entschieden ob das linke oder rechte 24 Bit sample auf den DACDAT Eingang gesendet werden soll. Das Digital Audio Bit Clock (BCLK) Signal gibt den Clock an mit dem die einzelnen Bits gesendet werden sollen.

Der *Audio and Video Config Core*, konfiguriert den Codec mit denn gewünschten Einstellungen. Über ein I2C control Interface werden die Kontrollregister des Codec beschrieben. Für die Anwendung in diesem Projekt wird der *filt\_out* Ausgang mit einer Sampling Rate von 48 kHz verwendet. Zudem wird der Left Justified mode des Codec verwendet. In diesem Modus wird das erste Bit welches über den DACDAT Eingang eingelesen wird als MSB betrachtet [11].

Die IP Cores *Audio* und *Audio and Video Config* verlangen eine Clock Frequenz basierend auf der sampling Rate des Codec. Der *Audio clock Core* generiert diese verlangte Clock Frequenz. Diese beträgt 12.288 MHz [12].

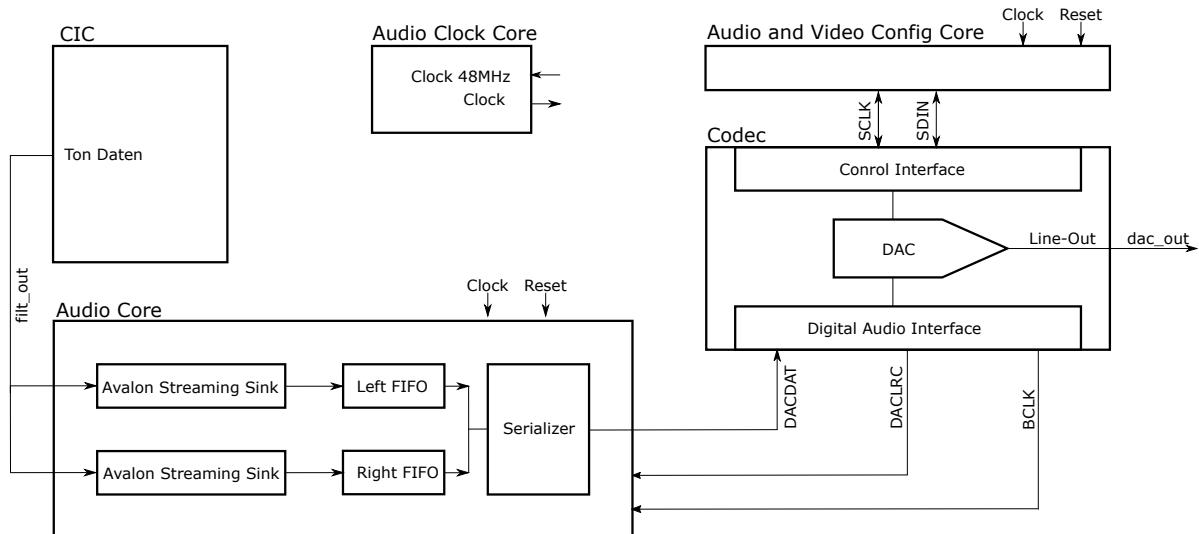


Abbildung 3.4: Blockschaltbild IP Cores, Konfiguration für die Verwendung des Codec

## 4 Validierung

In diesem Kapitel wird zuerst das Antennenoszillator PCB getestet. Anschliessend wird mit Simulationen des VHDL Codes, durch berechnen der jeweiligen Spektren der Zwischenresultate, deren gesamte Funktion getestet. Zum Schluss wird auf die Inbetriebnahme und das Debugging Bezug genommen.

### 4.1 Inbetriebnahme PCB

Die Inbetriebnahme des Antennenoszillator PCB wurde mit angeschlossener Antenne durchgeführt. Zuerst wurde die Komparator Speisung 3.3 V und die Oszillator Speisung 12 V überprüft. Der Ausgang *osc\_out* wurde mit dem KO gemessen. Wie gewünscht wird vom Antennenoszillator PCB ein Rechtecksignal erzeugt. Dieses hat, wie in Abbildung 4.1 zu sehen ist, eine Amplitude von 3.367 V und eine Frequenz von 562.3 kHz.

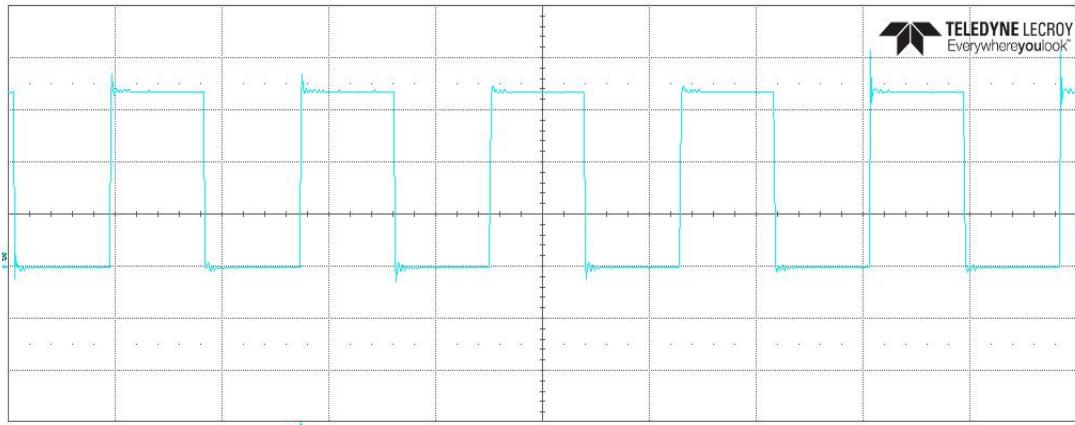


Abbildung 4.1: an *osc\_out* gemessenes Audiosignal des Theremin (1 V/div 1 us/div)

### 4.2 Simulation

Für die Bilder 4.2 bis 4.6 wurde der Reihe nach bei *ref\_out*, *mix\_out* und *filt\_out* (siehe Abbildung 2.2) aus der Simulation mit Questa Sim Werte entnommen. Es wurde die Methode aus dem Paper von Hanspeter Schmid verwendet um das Spektrum der Signale zu berechnen [13]. Da die simulierten Werte nicht Spannungswerte sind, haben die y-Achsen der Grafiken keine Einheit und die Werte selber sind nicht repräsentativ. Jedoch sind die Verhältnisse der Werte von Interesse, welche korrekt sind.

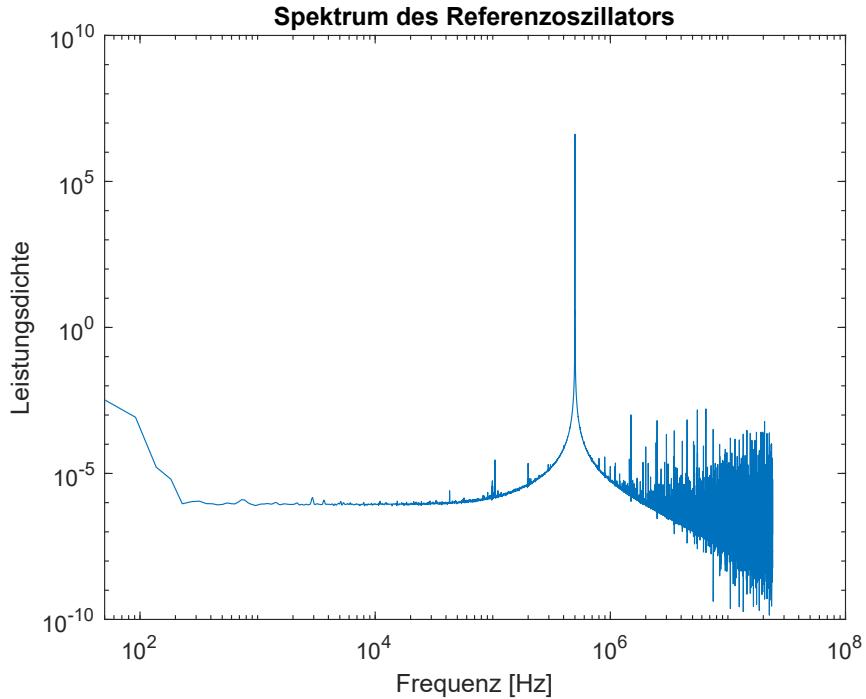
Abbildung 4.2 zeigt den Ausgang des Referenzoszillators *ref\_out*. Wie man sieht gibt es einen Peak der bei 500 kHz liegt, welcher dem gewünschten Signal entspricht. Zudem gibt es im DC Bereich einen Peak welcher 90 dB tiefer liegt als das gewünschte Signal. Dieser ist wohl darauf zurückzuführen, dass der Zahlenbereich, mit dem die Werte des Signals berechnet werden, mehr negative als positive Zahlen hat und das Signal einen kleinen negativen Offset hat. Sonstige Peaks liegen noch weiter unterhalb der zuvor genannten Frequenzen.

In Abbildung 4.3 sieht man das Resultat aus der Mischung des generierten Referenzoszillators und des in der Simulation generierten Rechtecksignals, welches an *mix\_out* ausgegeben wird. Die Frequenz des Rechtecksignals wurde dabei auf 501 kHz eingestellt. Dies bedeutet eine Differenz von 1 kHz. Wie man sieht sind die grössten Peaks im Spektrum bei 1 kHz und 1.001 MHz.

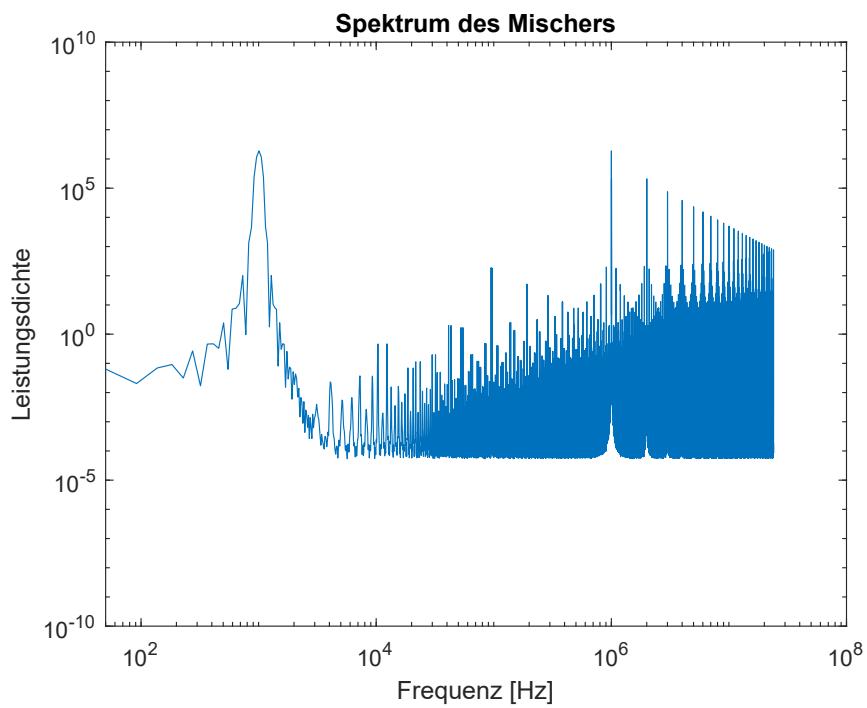
Dies entspricht der Formel 2.3. Die weiteren Peaks, welche mit höherer Frequenz immer mehr abschwächen, sind die Resultate der Mischung zwischen dem Signal des Referenzoszillators und den Oberwellen des Rechtecksignals. Der 1 kHz Peak erscheint nur deshalb breiter als die höheren Frequenzen, da das Spektrum doppelt logarithmisch aufgezeichnet wurde.

Abbildung 4.4 zeigt das Resultat der Filterung des gemischten Signals an *filt\_out*. Wie man sieht hatte das CIC-Filter nicht nur die Filterung des Signals zur Folge. Die Abtastfrequenz wurde zudem von 48 MHz auf 48 kHz verkleinert, um das Signal am Codec auszugeben.

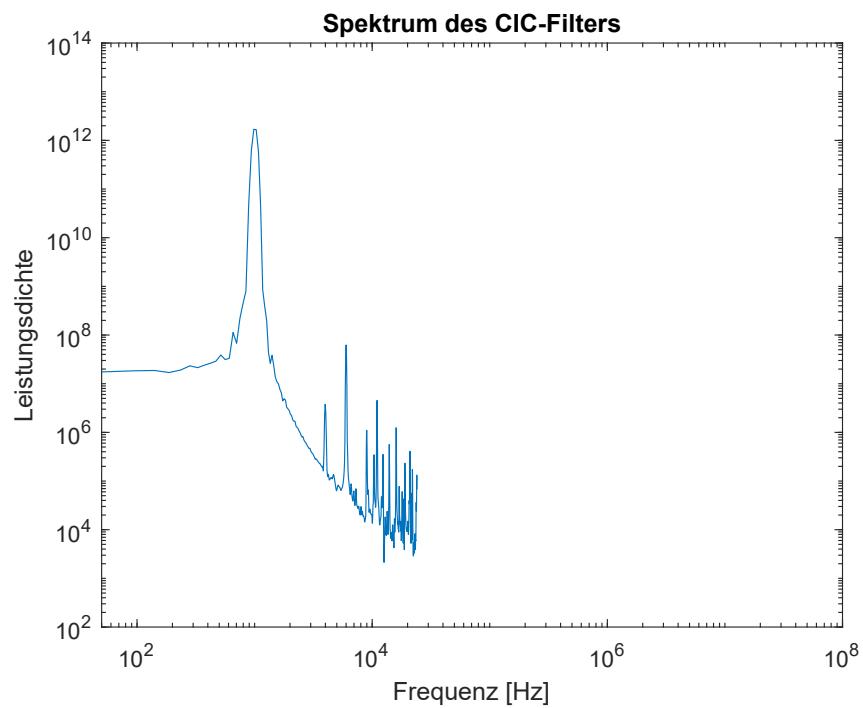
Zuletzt zeigen die Abbildungen 4.5 und 4.5 etwas genauer, dass das CIC-Filter verschiedenste Effekte auf das Signal hatte. Zuerst sieht man sehr schön die zuvor erwähnte Unterabtastung des Signals. Weiter ist ersichtlich, dass das Signal um 150 dB verstärkt wurde. Dies ist einerseits aufgrund der Funktionsweise des CIC-Filters entstanden und andererseits der Vergrößerung der Bitanzahl der Werte für die Übertragung an den Codec. Das Filter Vergrößert die Werte von 16 Bit Zahlen zu 26 Bit zählen. Um diese dann an den Codec weiterzugeben werden die Werte von 26 Bit auf 32 Bit erhöht. Weiter ist ersichtlich, dass das CIC-Filter Aliasing hervorgerufen hat, welches höherfrequente Peaks in tiefere Bereiche verschoben hat.



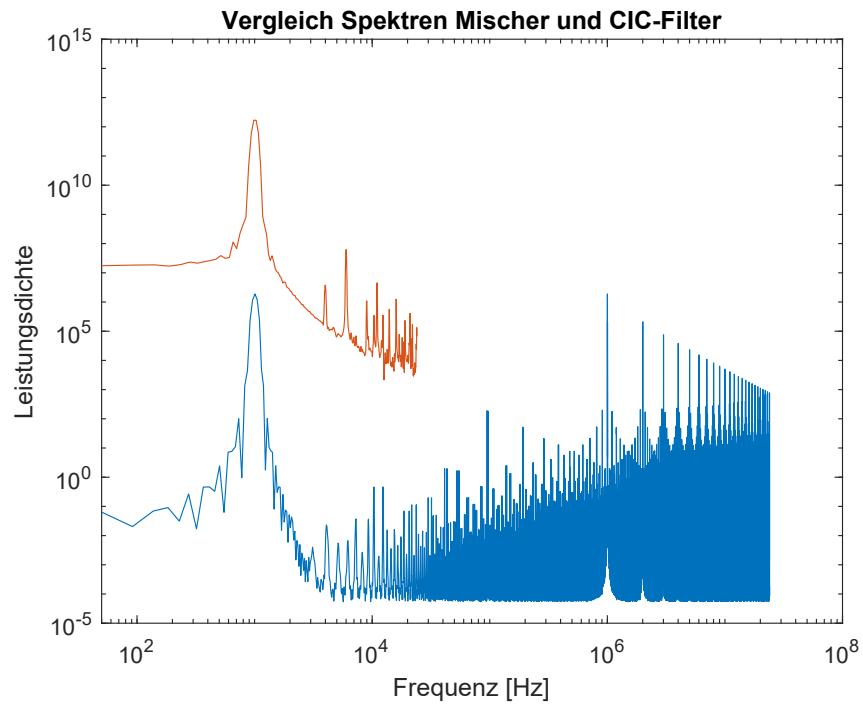
**Abbildung 4.2:** Spektrum am Ausgang des Referenzoszillators mit 16 Bit Werten aus Simulation (gemessen an *ref\_out*)



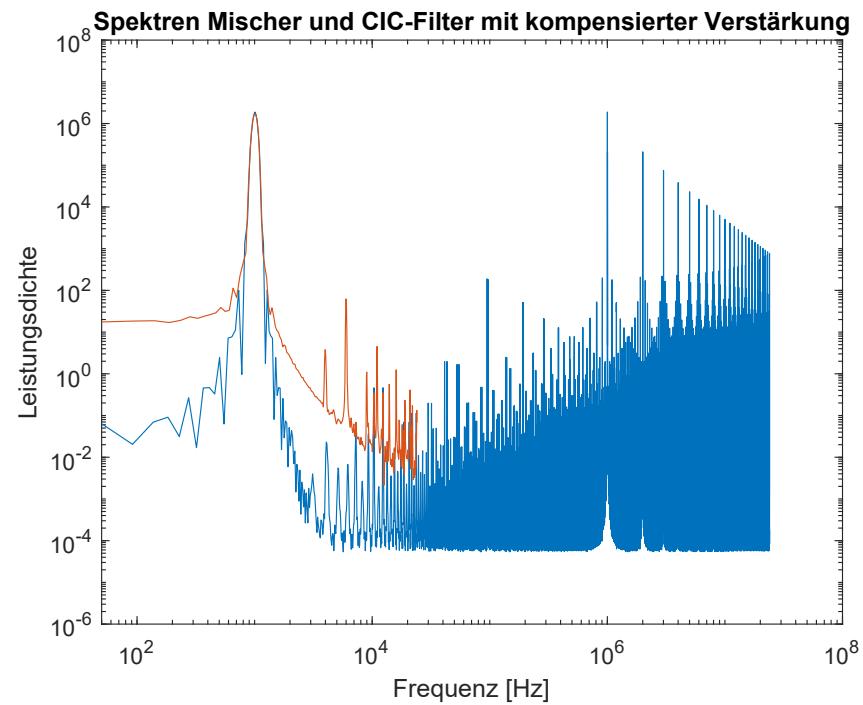
**Abbildung 4.3:** Spektrum am Ausgang des Mischers mit 16 Bit Werten aus Simulation (gemessen an mix\_out)



**Abbildung 4.4:** Spektrum am Ausgang des Tiefpassfilters mit 26 Bit Werten aus Simulation (gemessen an cic\_out)



**Abbildung 4.5:** Vergleich der Spektren an den Ausgängen des Mischers (16 Bit Werte) und des Tiefpassfilters (26 Bit Werte) ohne Kompensierung der Verstärkung des Filters (gemessen an cic\_out und mix\_out)



**Abbildung 4.6:** Vergleich der Spektren an den Ausgängen des Mischers (16 Bit Werte) und des Tiefpassfilters (26 Bit Werte) mit Kompensation der Verstärkung des Filters (gemessen an cic\_out und mix\_out)

### 4.3 Debugging

Um das SRAM Object File (SOF) zu erstellen und auf das Board zu laden wurde Quartus Prime von Intel verwendet. Bild 4.7 zeigt den verwendeten Testaufbau.

Dabei fiel auf, dass das Theremin zwar spielbar ist, jedoch ist ein Knistern zu hören. Bild 4.8 zeigt eine Aufnahme mit dem Oszilloskop des Audiosignals. Die erste Annahme war, dass es Probleme mit der Kommunikation mit dem Codec gibt. Aus diesem Grund wurde zur Bestätigung ein Rechtecksignal mit der Frequenz 1 kHz an *mix\_out* angelegt und geschaut ob der Fehler immer noch besteht. Wie man aus Bild 4.9 sieht besteht der gleiche Fehler bei diesem Test. Somit kann der Fehler auf die Kommunikation zum Codec zurückgeführt werden. Die Veränderung der Rechteckform ist auf die Filterung des Signals durch das CIC-Filter zurückzuführen. Aus Zeitgründen und weil diese Komponente später im P6 anders realisiert wird, wurde dieser Fehler jedoch ohne Korrektur belassen.

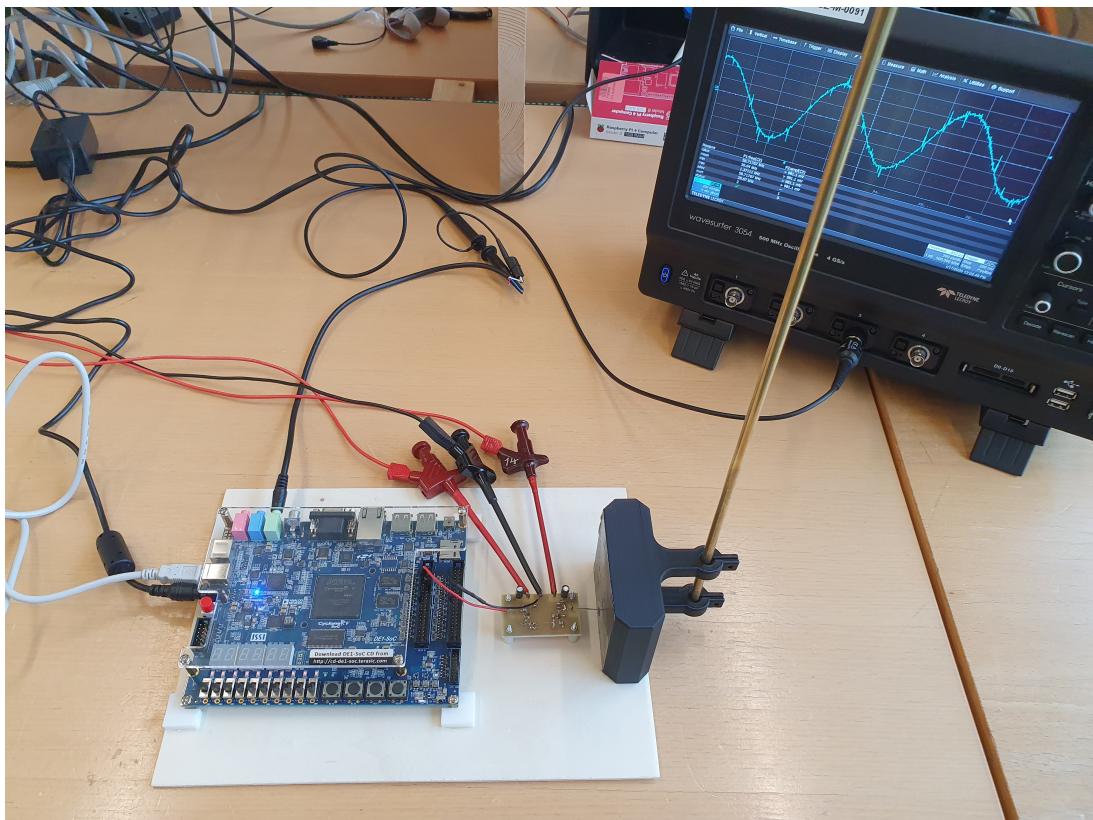


Abbildung 4.7: Verwendeter Testaufbau

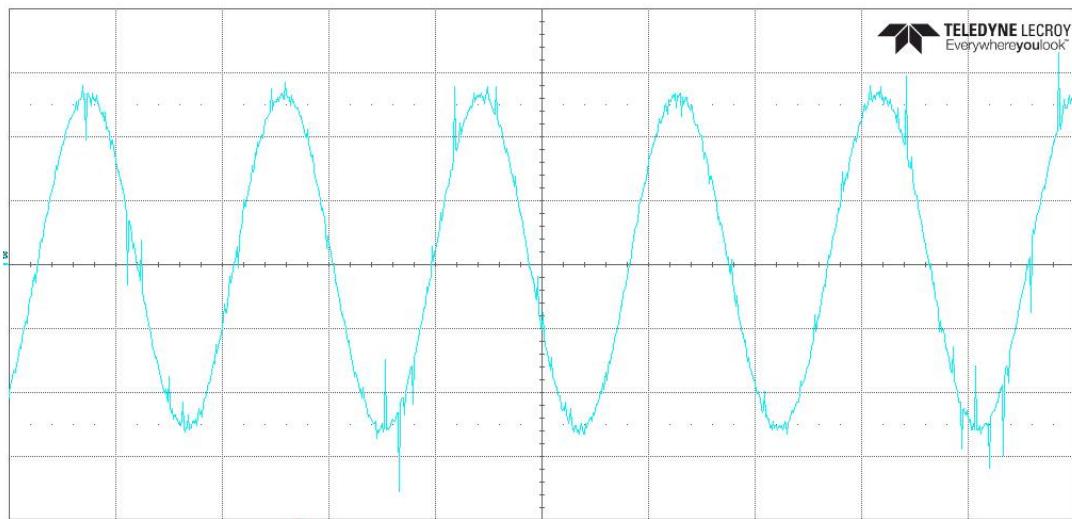


Abbildung 4.8: gemessenes Audiosignal des Theremin (200 mV/div 500 us/div)

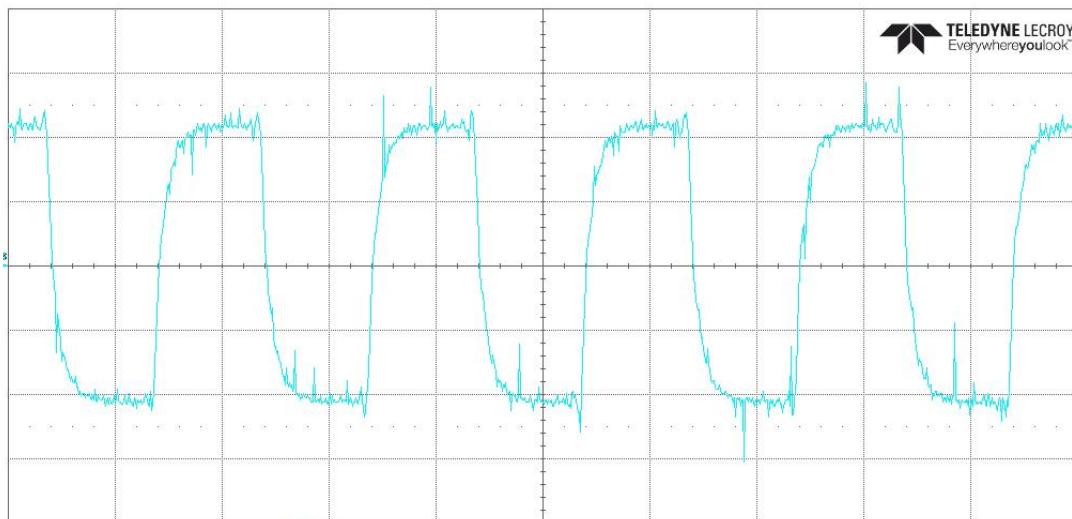


Abbildung 4.9: gemessenes Audiosignal mit generiertem Rechtecksignal (200 mV/div 500 us/div)

## 5 Schlusswort

Im Rahmen des Projekt 5 wurde eine digitale Plattform für die Verarbeitung von Signalen einer Thereminantenne entwickelt. Alle Komponenten ausser der Antennenoszillator wurden in VHDL realisiert. Die VHDL Komponenten wurden so realisiert, dass diese im Projekt 6 weiter gebraucht werden können. Momentan lässt sich das Theremin ohne Lautstärkeantenne spielen. Über zwei Taster kann der digitale Referenzoszillator manuell auf die Frequenz des Tonhöhenoszillators abgestimmt werden. Sobald das Theremin kalibriert ist kann es Töne von ca. 100-2000Hz spielen. Die Ziele welche in der Projektklärung definiert wurden konnten erfüllt werden. Bei der kontinuierlichen Tongenerierung gibt es noch eine Unschönheit bei der Ansteuerung des Codec. Es ist im generierten Ton ein Knacken zu hören, welches auf einen Fehler in der Ansteuerung des verwendeten Codec zurückzuführen ist. Dieser Fehler besteht nach wie vor. Jedoch wird diese Ansteuerung in Projekt 6 sowieso anders realisiert.

Im Projekt 6 wird die zweite Antenne implementiert, um gleichzeitig die Lautstärke einzustellen zu können. Des weiteren soll es möglich sein diskrete Töne zu spielen. Dieser Modus soll es Anfängern ermöglichen bekannte Melodien nachspielen zu können.

Damit das theoretische Wissen aus dem Fach digitale Schaltungstechnik (dst) in die Praxis umgesetzt werden kann, wird ein Nios Soft Core Prozessor implementiert. Dieser übernimmt die Ansteuerung des Codec und die Modus Verwaltung. Beim starten des Theremin soll ein automatisches Tuning des Referenzoszillators stattfinden. Dazu wird der digitale Referenzoszillator auf die Frequenz des Antennenoszillators abgestimmt. Um das Theremin für Messen zu verwenden wird das DE1-SoC Board und die Antennenoszillatoren mit den Antennen in ein ansprechendes Gehäuse verbaut. Die Antennen sollen abgeschraubt werden können um einen komfortablen Transport zu ermöglichen.

Als erstes wird im Projekt 6 mit der Implementierung der Lautstärkeantenne auf dem FPGA und dem Redesign des PCB begonnen. Zudem muss Recherche in das Thema Nios Soft Core Prozessor angestellt werden, um diesen später implementieren zu können.

## 6 Ehrlichkeitserklärung

Mit der Unterschrift bestätigen die Unterzeichnenden Teammitglieder, dass die vorliegende Projektdokumentation selbstständig im Team und ohne Verwendung anderer, als der angegebenen Hilfsmittel verfasst wurde, sämtliche verwendeten Quellen erwähnt und die gängigen Zitierregeln eingehalten wurden. Eine Überprüfung der Arbeit auf Plagiate mithilfe elektronischer Hilfsmittel darf vorgenommen werden.

Unterschrift:

Ort, Datum:

Unterschrift:

Ort, Datum:

## Literatur

- [1] WDR. (2010). 05. August 2010 - Vor 90 Jahren: Musikinstrument Theremin wird vorgestellt. (Abrufdatum 17.01.2020), Adresse: <https://www1.wdr.de/stichtag/stichtag5844.html>.
- [2] Garrett Tiedemann. (2016). A history of the theremin in movie music. (Abrufdatum 17.01.2020), Adresse: <https://www.classicalmpr.org/story/2016/07/15/theremin-movie-music>.
- [3] T. Riegler, „Theremin zum selberbauen“, *Franzis*, Jg. 73, Nr. 1, S. 1–73, 2018. DOI: GTIN4019631670519.
- [4] Kenneth D. Skeldon, Lindsay M. Reid, Viviene McInally, Brendan Dougan, and Craig Fulton. (1998). Physics of the Theremin. (Abrufdatum 03.01.2020), Adresse: <https://pdfs.semanticscholar.org/159b/8f7ab33083fc1b8de584ec338b0ee2f6fd7b.pdf>.
- [5] Esteban O. Garcia. (2006). Paper Cordic Generator. (Abrufdatum 17.01.2020), Adresse: <https://ccc.inaoep.mx/~rcumplido/papers/2006-Garcia-Pipelined%20CORDIC%20Design.pdf>.
- [6] Eugene B. Hogenauer. (1981). An Economical Class of Digital Filters for Decimation and Interpolation. (Abrufdatum 17.01.2020), Adresse: <http://read.pudn.com/downloads163/ebook/744947/123.pdf>.
- [7] Intel. (2007). Understanding CIC Compensation Filters. (Abrufdatum 17.01.2020), Adresse: <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an455.pdf>.
- [8] Analog Devices. (2020). LTC6752/LTC6752-1/LTC6752-2/LTC6752-3/LTC6752-4. (Abrufdatum 03.01.2020), Adresse: <https://www.analog.com/media/en/technical-documentation/data-sheets/6752fc.pdf>.
- [9] Wolfson. (2012). Portable Internet Audio CODEC WM8731. (Abrufdatum 17.01.2020), Adresse: [https://statics.cirrus.com/pubs/proDatasheet/WM8731\\_v4.9.pdf](https://statics.cirrus.com/pubs/proDatasheet/WM8731_v4.9.pdf).
- [10] Intel. (2017). Audio core for Intel DE-Series Boards. (Abrufdatum 17.01.2020), Adresse: [ftp://ftp.intel.com/Pub/fpgaup/pub/Intel\\_Material/18.0/University\\_Program\\_IP\\_Cores/Audio\\_Video/Audio.pdf](ftp://ftp.intel.com/Pub/fpgaup/pub/Intel_Material/18.0/University_Program_IP_Cores/Audio_Video/Audio.pdf).
- [11] —, (2017). Audio/Video Configuration Core. (Abrufdatum 17.01.2020), Adresse: [ftp://ftp.intel.com/Pub/fpgaup/pub/Intel\\_Material/15.0/University\\_Program\\_IP\\_Cores/Audio\\_Video/Audio\\_and\\_Video\\_Config.pdf](ftp://ftp.intel.com/Pub/fpgaup/pub/Intel_Material/15.0/University_Program_IP_Cores/Audio_Video/Audio_and_Video_Config.pdf).
- [12] —, (2014). Clocks for Altera DE-Series Boards. (Abrufdatum 17.01.2020), Adresse: [ftp://ftp.intel.com/Pub/fpgaup/pub/Intel\\_Material/15.0/University\\_Program\\_IP\\_Cores/Clocks/Altera\\_UP\\_Clocks.pdf](ftp://ftp.intel.com/Pub/fpgaup/pub/Intel_Material/15.0/University_Program_IP_Cores/Clocks/Altera_UP_Clocks.pdf).
- [13] Hanspeter Schmid. (2012). How to use the FFT and Matlab's pwelch function. (Abrufdatum 17.01.2020), Adresse: <http://www.schmid-werren.ch/hanspeter/publications/2012fftnoise.pdf>.

## A Projektklärung



Fachhochschule Nordwestschweiz  
Hochschule für Technik

# digitales Theremin

## Projektklärung

HS19, Projekt 5

Windisch, 11.10.2019

Auftraggeber: Herr Prof. Dr. Hanspeter Schmid

Betreuung: Herr Prof. Dr. Hanspeter Schmid  
Herr Prof. Karl Schenk

Team: Andreas Frei  
Dennis Aeschbacher

Studiengang: Elektro- und Informationstechnik (EIT)

## Inhaltsverzeichnis

1	Übersicht.....	3
1.1	Ausgangslage.....	3
1.2	Ziele P5.....	3
1.3	Ziele P6.....	4
2	Sensor Evaluation.....	5
3	Konzept.....	6
3.1	Komponenten P5.....	6
3.2	Komponenten P6 .....	7
3.3	Blockschaltbild .....	8
4	Risiken .....	9
5	Projektstrukturplan .....	10
6	Quellenverzeichnis .....	10

# 1 Übersicht

## 1.1 Ausgangslage

Ein Theremin ist ein Musikinstrument, das mit "den Händen in der Luft" gespielt wird (<https://www.youtube.com/watch?v=IY7sXKGZl2w>). Es kommt in vielen Musikstücken, aber auch in Filmen (z.B. "Inspector Barnaby"-Serie, Hellboy <https://www.youtube.com/watch?v=x1cekAOKH5E>) zum Einsatz.

Beim Theremin beeinflusst die elektrische Kapazität des menschlichen Körpers ein elektromagnetisches Feld. Dabei steuert die Position der Hände gegenüber zwei Antennen die Stärke der Veränderung. Die sich ändernde Schwingung des Feldes wird verstärkt und als Ton über einen Lautsprecher ausgegeben [1].

Ziel des Projekts ist, basierend auf einem Bausatz eines analogen Theremins (z.B. Franzis) ein digitales Theremin zu bauen. Dabei soll die gesamte Signalverarbeitung in einem FPGA realisiert werden was zum Beispiel den Ausgleich von Lautsprecherfrequenzgängen, die direkte digitale Aufzeichnung, sowie weitere Soundeffekte erlaubt. Das digitale Theremin soll in ein kompaktes ansprechendes Gehäuse gepackt werden.

Um diese Ziele zu erreichen wird die Arbeit über das fünfte und das sechste Semester verteilt. Im fünften Semester soll eine digitale Plattform für die Ansteuerung und Kalibrierung der Tonhöhenantenne entwickelt werden. Dazu wird ein FPGA Entwicklungsboard verwendet. Im sechsten Semester wird zusätzlich die Lautstärkeantenne und verschiedene Effekte implementiert.

## 1.2 Ziele P5

Die unten aufgelisteten Ziele sollen im 5. Semester erreicht werden, um die Grundlage des Theremins abzuschliessen.

Nr.	Ziel	Inhalt
1	Beschränkung auf FPGA	Alle Komponenten ausser die Signalgenerierung (Antenne, Schwingkreis und Audioverstärker) werden im FPGA in VHDL realisiert.
2	kontinuierliche Tongenerierung	Man kann das Theremin spielen jedoch ohne Lautstärke Antenne.
3	Weiterarbeit beachten	Die Arbeit soll so gestaltet werden, dass sie im Projekt 6 weitergeführt werden kann.
4	Manuelles Tuning der Tonhöhe	Der digitale Referenzoszillator kann manuell auf die Frequenz des Tonhöhenoszillators abgestimmt werden.
5	Vier Oktaven spielbar	Das Theremin kann Töne von 130.813 (c) bis 2.093kHz (C'') kontinuierlich spielen.

### 1.3 Ziele P6

Die nächsten Ziele werden erst im 6. Semester angegangen und legen besonderes Augenmerk auf zusätzliche Features und den «Eye Catcher» Faktor.

Nr.	Ziel	Inhalt
1	Lautstärke Antenne	Zweite Antenne implementieren, um gleichzeitig Lautstärke einstellen zu können
2	Diskrete Tongenerierung	Möglichkeit diskrete Töne zu Spielen (Bsp: Tonleiter, Pentatonische Tonleiter)
3	Zusätzliche Effekte	Implementierung zusätzlicher Effekte (Bsp:Sampling Effekte)
4	ansprechendes Gehäuse	Ein ansprechendes Gehäuse bauen wo die Elektronik ersichtlich ist.
5	Menu auf LCD Display	Das Theremin soll über ein LCD Display bedienbar sein. Dies würde über den uC auf dem FPGA gesteuert werden.
6	Automatisches Tuning der Tonhöhe	Der digitale Referenzoszillator wird automatisch auf die Frequenz des Tonhöhenoszillators abgestimmt.

## 2 Sensor Evaluation

Beim klassischen Theremin werden für das berührungslose Spielen zwei Antennen verwendet. Es gibt jedoch noch andere Möglichkeiten Abstände und Bewegungen zu detektieren. In diesem Abschnitt werden die Vor- und Nachteile von drei Sensoren aufgelistet und evaluiert.

Rangfolge	Technologie	Nachteile	Vorteile
1	Antennen	-Der Schwingkreis der Antenne muss auf einem PCB realisiert werden	-Durch den menschlichen Körper wird die Kapazität des Schwingkreises beeinflusst. Das Theremin kann durch feine Handbewegungen beeinflusst werden. Dies führt dazu das das Gerät wie ein klassisches Theremin gespielt werden kann [1].
2	Ultraschallsensor	-Plane Objekte werden am besten detektiert. Der Bewegungsgrad der Hand des Spielers wird somit eingeschränkt [2]. -Das Gerät könnte nicht wie ein Theremin gespielt werden, da nur Distanzen detektiert werden können.	-Braucht keine zusätzliche Elektronik für die Verarbeitung.
3	ToF-Kamera	-Für jede Bewegung und Distanz müsste ein Ton festgelegt werden. Dies würde das experimentelle musizieren sehr stark einschränken [3]. -Die technische Realisierung ist gegenüber den anderen Sensoren sehr aufwändig.	

Die kurzgehaltene Recherche über mögliche Sensoren hat ergeben, dass die Antennen am besten geeignet sind als Sensoren. Durch die Antennen verliert das Theremin nicht sein klassisches Aussehen und seine Spielweise. Nur mit den Antennen und der ToF-Kamera ist es möglich kontinuierliche Töne zu generieren. Der Aufwand mit einer ToF-Kamera kontinuierliche Töne zu generieren ist sehr zeitintensiv und passt nicht in den Zeitrahmen des Projektes. Daher wurde die Antenne als Sensor für das Theremin gewählt.

## 3 Konzept

Die unten beschriebenen Komponenten können in Kapitel 3.3 Blockschaltbild gefunden werden.

### 3.1 Komponenten P5

#### **Tonhöhenoszillator**

In dem Block Tonhöhenoszillator kann die Frequenz des Ausgangssignals verändert werden, indem die Distanz zwischen dem Spieler und der angeschlossenen Antenne verändert werden. Dies, da der Spieler des Instruments das elektrische Feld der Antenne so verändert, dass die Frequenz des Oszillators leicht ändert. Dieser Teil wird nicht im FPGA sondern analog realisiert.

#### **Referenzoszillator**

Der Referenzoszillator wird benötigt, um das Signal des Tonhöhenoszillators im Mischer zu bearbeiten. Um diesen zu realisieren können zwei Methoden angewandt werden. Einerseits die Verwendung von Lookup Tables (LUTs) und andererseits der Einsatz des Cordic Algorithmus.

#### **Mischer**

Die Komponente Mischer ist dafür zuständig die Frequenz des Tonhöhenoszillators, welche nicht im hörbaren Bereich ist in den hörbaren Bereich zu bringen. Dabei multipliziert er die Signale des Tonhöhenoszillators mit der des Referenzoszillators. Das Resultat sind zwei Signale, welche einerseits die Summe der beiden Frequenzen vor dem Mischer haben und andererseits die Differenz der beiden Signale.

#### **Tiefpassfilter**

Da der zuvor erwähnte Mischer auch ein hochfrequentes Signal generiert, muss das gemischte Signal noch mittels eines Tiefpassfilters gefiltert werden.

#### **DAC/ADC**

Hier sind die Übergänge zwischen den analogen Signalen und dem FPGA.

## 3.2 Komponenten P6

### Lautstärkeoszillator

Für den Lautstärkeoszillator und dessen Verarbeitung wird voraussichtlich derselbe Aufbau wie beim Tonhöhenoszillator eingesetzt. Dabei entspricht die Höhe der Frequenz am Ausgang des Tiefpasses der Höhe der Lautstärke.

### Lautstärkeeinstellung

Die Lautstärke wird in Funktion der Frequenz des Lautstärkeoszillators verändert. Dabei wird die gemessene Frequenz in einen Zahlenwert umgewandelt, welche die Lautstärke einstellt.

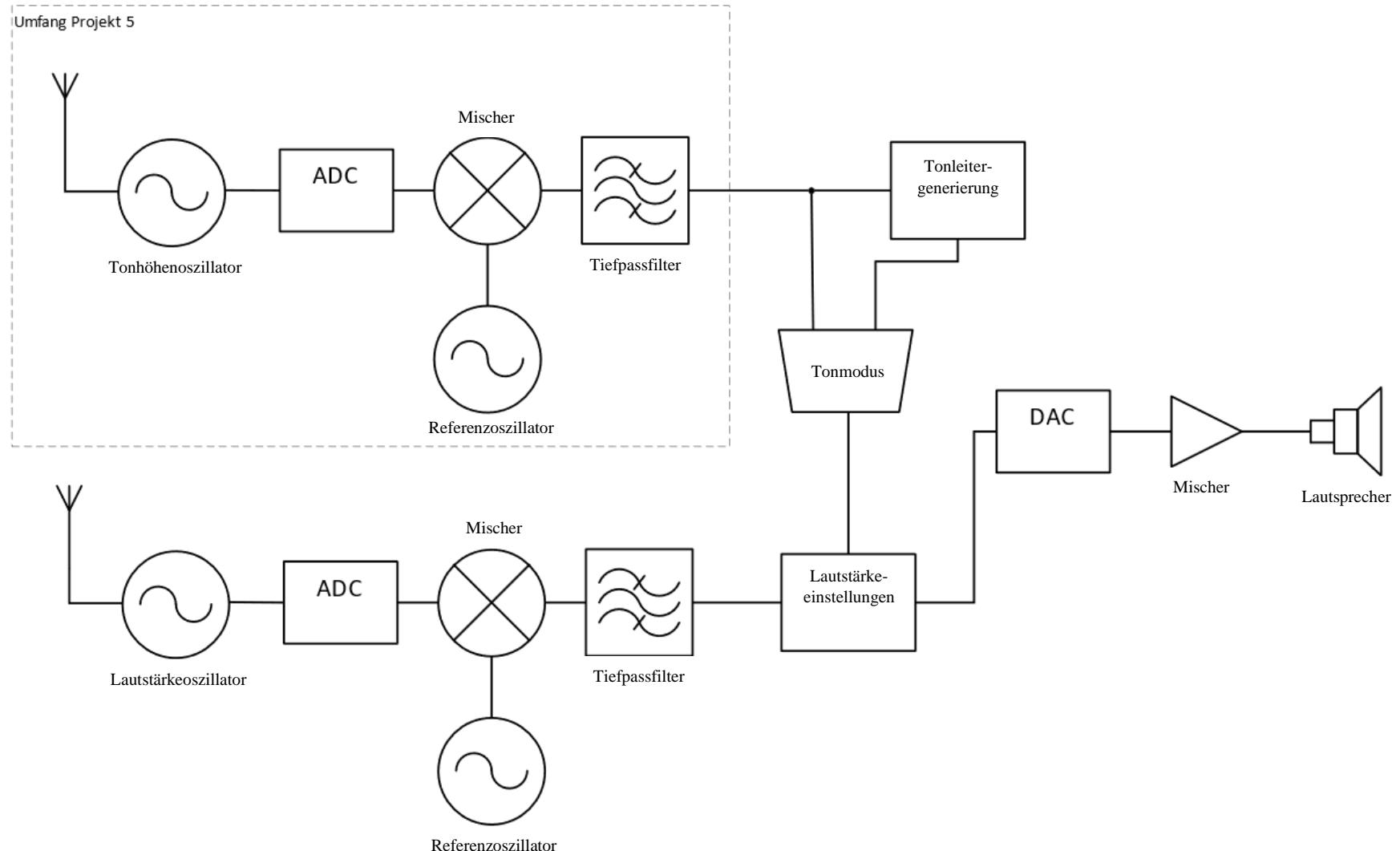
### Tonleitergenerierung

Um das Spielen des Theremins zu vereinfachen kann man auf den Tonleitermodus wechseln, womit man beispielsweise die pentatonische Tonleiter spielen kann. Somit würden diskrete Töne gespielt und nicht kontinuierliche, was es vereinfacht das Instrument zu bedienen. Dies soll auch ungeübten Spielern ermöglichen etwas zu spielen. Zusätzlich soll es möglich sein, Töne anderer Instrumente zu spielen

### Verstärker und Lautsprecher

Im Projekt 5 wird vorerst der Audioausgang des Entwicklungsboards verwendet, um die Funktionalität zu testen. Im Projekt 6 wird dann abgeklärt ob selbst ein Verstärker entworfen werden soll.

### 3.3 Blockschaltbild



## 4 Risiken

Gewicht	Beschreib	Konsequenzen	Prävention
1	Die Mischung und Filterung des Signals funktioniert nicht.	Das Theremin kann nicht gespielt werden	Das Mischen in einem zeit und wertdiskretem Simulations- Tool simulieren.
2	digitaler Referenzoszillator funktioniert nicht	Das Theremin kann nicht gespielt werden.	Der digitale Referenzoszillator in einem zeit und wertdiskretem Simulations- Tool simulieren.
3	Die Dynamik des analogen Antennen Teils reicht nicht aus, um eine Differenz von bis zu 2kHz zu generieren.	Das Theremin hat nicht die volle Dynamik von 4 Oktaven(2kHz).	Ausmessen des bestehenden Franzis Bausatzes.
4	Inbetriebnahme des ARM-based hard processor system(HPS) für die Ansteuerung eines LCD	Ist es nicht möglich die Inbetriebnahme des HPS zu machen, schränkt dies stark die Bedienerfreundlichkeit des Theremins ein.	Bereits bestehende Projekte konsultieren

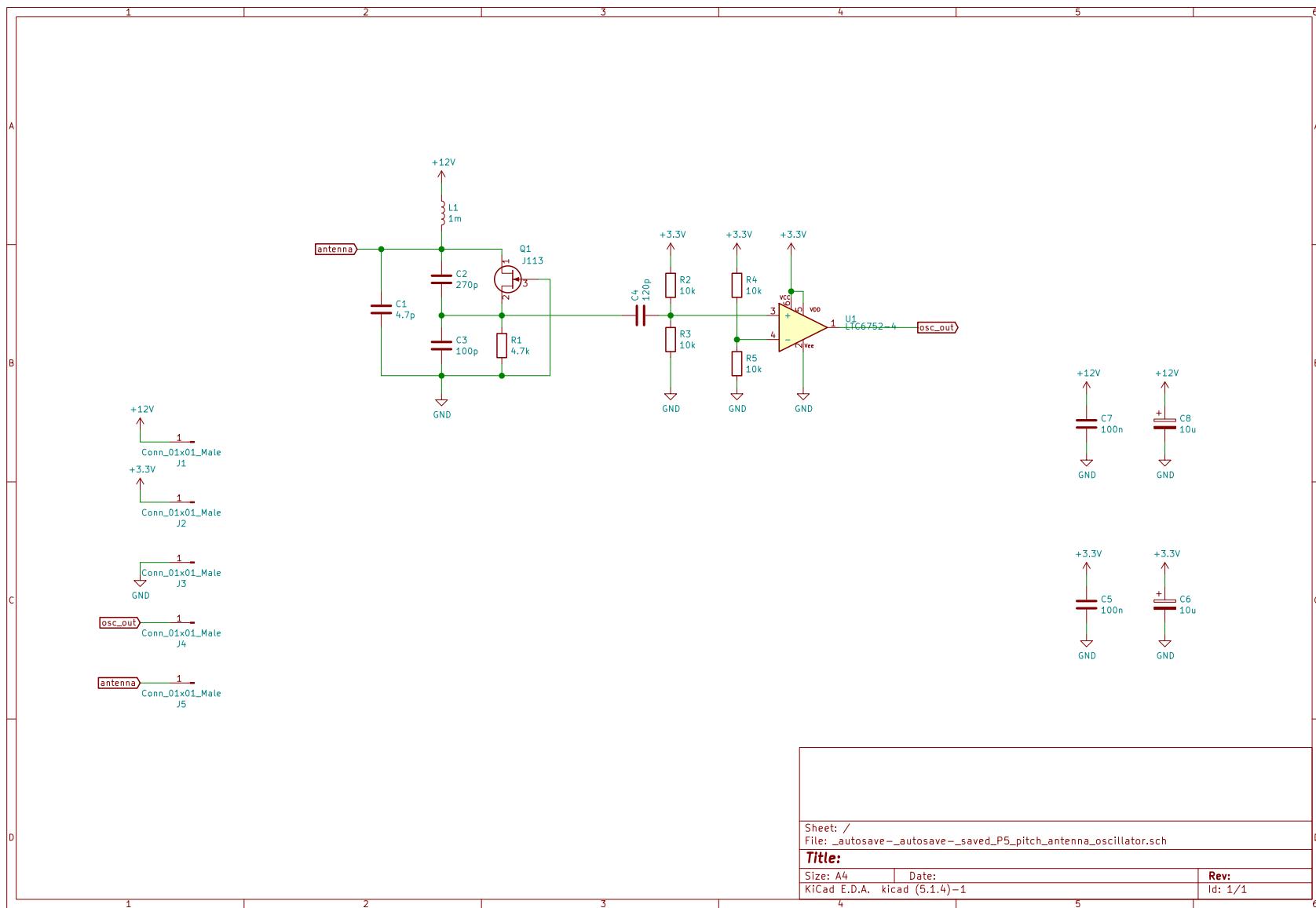
## 5 Projektstrukturplan

Der Projektstrukturplan wurde auf Grund der Übersichtlichkeit im Excel File Projektstrukturplan.xlsx realisiert.

## 6 Quellenverzeichnis

- [1] T.Riegler, "Wie funktioniert ein Theremin? ,“ Theremin zum selberbauen. München, Franzis Verlag, 2018, 12. [Online]  
Available:[https://files.elv.com/Assets/Produkte/14/1448/144855/Downloads/65347\\_Theremin\\_selber-bauen\\_Druckdaten.pdf](https://files.elv.com/Assets/Produkte/14/1448/144855/Downloads/65347_Theremin_selber-bauen_Druckdaten.pdf) [09.10.2019]
- [2] Ege Elektronik, "Ultraschallsensoren Technik und Anwendung,“ 2017. [Online]  
Available:[https://ege-elektronik.com/de/produkte/temperatur-druck-ultraschall/anwendung\\_ultraschall.html](https://ege-elektronik.com/de/produkte/temperatur-druck-ultraschall/anwendung_ultraschall.html) [09.10.2019]
- [3] "Time-of-flight camera ,“ Wikipedia, 2019. [Online]  
Available: [https://en.wikipedia.org/wiki/Time-of-flight\\_camera](https://en.wikipedia.org/wiki/Time-of-flight_camera)

## B Schema Antennenoszillator



## C PCB Antennenoszillator

