Fachhochschule Nordwestschweiz
Hochschule für Technik

digitales Theremin
Fachbericht

Projekt 6
16. 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:	Frühlingssemester 2020

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

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 Messeobjekt 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 5 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

bla bla

2.2 digitales Theremin

bla bla

3 Realisierung

Das digitale Theremin ist auf dem Entwicklungsboard DE1-SOC von terasIC aufgebaut. Dieses enthält ein Cyclone V 5CSEMA5 FPGA von Intel. Weiter befindet sich auf dem Board der Audio Codec WM8731 von Wolfson für die Ausgabe an einem Lautsprecher. In Abbildung 3.1 ist der Aufbau des Digitalen Theremin aufgezeigt inklusive der Peripherie ausserhalb des FPGA.
Das Theremin, welches im FPGA aufgebaut ist, besteht aus zwei Bereichen. Einerseits der Signalverarbeitung und Übermittlung an den Codec. Dieser besteht aus den Komponenten *Volume* und *Pitch Generation*, *DC-FIFO* und dem *Audio Serializer*. Der zweite Bereich ist Das Nios System. Dieses besteht aus dem Prozessor und diversen IP Cores, welche die Kommunikation mit den Peripherien ermöglicht. Ausserhalb des FPGA ist zudem das entwickelte PCB, welches die beiden Antennenoszillatoren enthält, mit welchen das Theremin gespielt werden kann.
Die Kommunikation zwischen dem Nios Prozessor und den anderen Komponenten geschieht über das *Avalon Memory Mapped Interface*. Der Prozessor ist in dieser Kommunikation Master und die restlichen Komponenten Slaves. Die Übertragung der Audioinformation in der Signalverarbeitung geschieht über das *Avalon Streaming Interface*. Wobei Sender als Streaming Source und Empfänger als Streaming Sink deklariert sind. Das Streaming Interface ist notwendig für den Einsatz des Dual-Clock-FIFO (DC-FIFO). Dieses übernimmt den Übergang verschiedener Clockregionen zwischen den Komponenten *Pitch Generation* und *Audio Serializer*.
Die Clocks, welche zu den verschiedenen Komponenten gehen sind in Abbildung 3.1 für eine bessere Überschaulichkeit weggelassen worden. Für eine Liste aller Clock Frequenzen und deren Ziel siehe Kapitel 3.2.

3.1 Antennenoszillator

Text überarbeiten Andy?

3.2 Clock

Die verschiednen Clocks für die Hardwarekomponenten und die CPU werden in zwei PLL Blöcken generiert. Ein Block für die Signalverarbeitung und einer für das Nios System. In Tabelle 3.1 sind alle Frequenzen aufgelistet.
Alle Frequenzen welche nicht 50MHz sind ergaben sich daraus, dass die externen Peripherien, welche mit den entsprechenden IP Cores verbunden sind, diese Frequenzen als Vorgabe haben. Weiter benötigen die Komponenten Pitch- und Volume Generation die Frequenz 54Mhz, da deren Frequenz ein vielfaches von 48kHz sein muss.

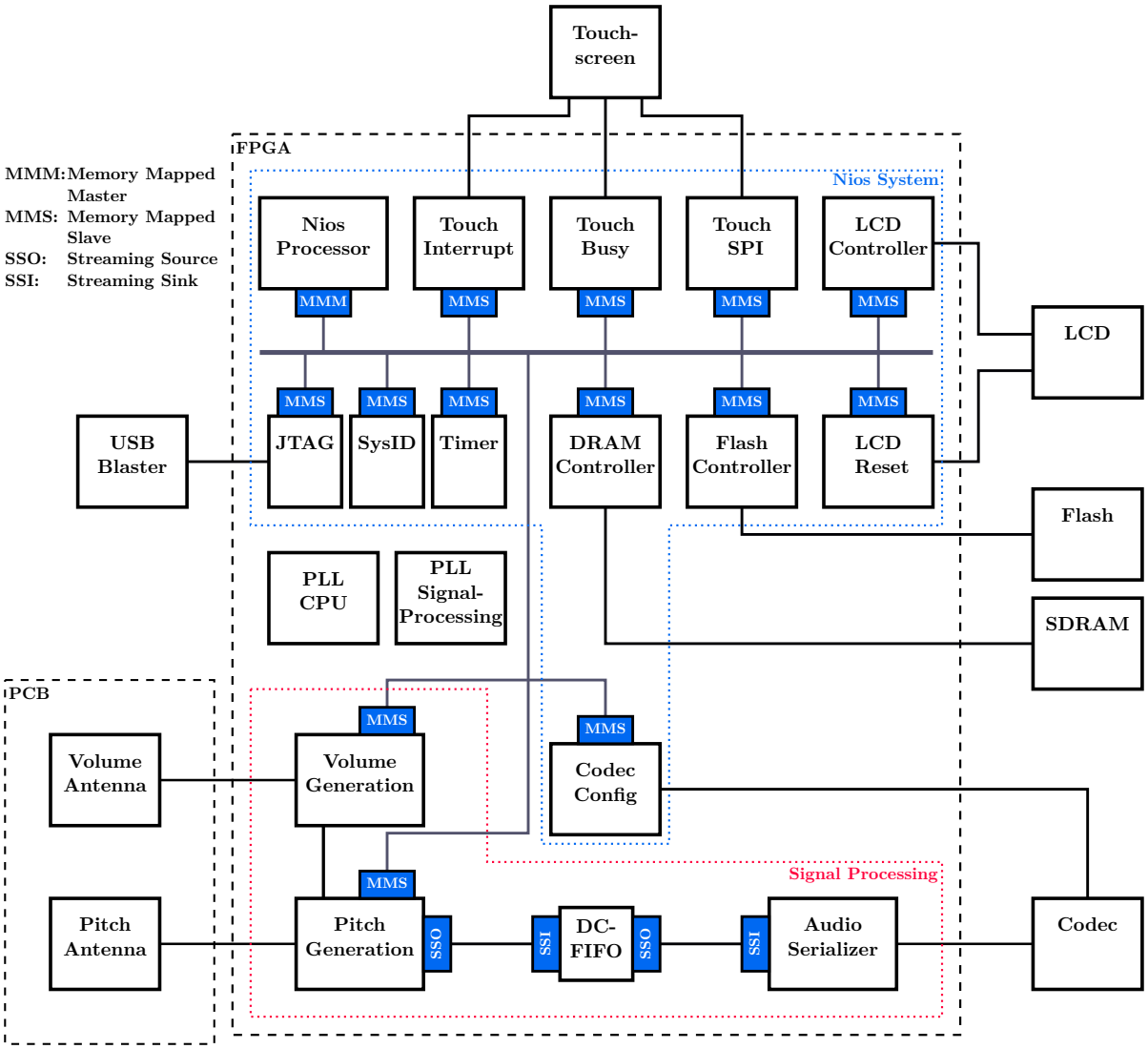


Abbildung 3.1: Blockschaltbild gesammtes Theremin

Tabelle 3.1: Clockfrequenzen der verschiedenen Komponenten

Komponente	Frequenz	PLL Core
Nios Processor	50 MHz	PLL CPU
JTAG Controller	50 MHz	PLL CPU
Timer	50 MHz	PLL CPU
SysID	50 MHz	PLL CPU
DRAM Controller	50 MHz	PLL CPU
SDRAM	50 MHz	PLL CPU
LCD Controller	15 MHz	PLL CPU
LCD Reset	15 MHz	PLL CPU
Touch Interrupt	15 MHz	PLL CPU
Touch Busy	15 MHz	PLL CPU
Touch SPI	15 MHz	PLL CPU
Audio Config	12 MHz	PLL CPU
Flash Controller	25 MHz	PLL CPU
Pitch Generation	54 MHz	PLL Signal-Processing
Volume Generation	54 MHz	PLL Signal-Processing
DC-FIFO Input	54 MHz	PLL Signal-Processing
DC-FIFO Output	24 MHz	PLL Signal-Processing
Audio Serializer	24 MHz	PLL Signal-Processing

3.3 CPU

Der eingesetzte Nios Prozessor ist für die Bedienung des Theremin und die Steuerung der Signalverarbeitungshardware zuständig. Die diversen eingesetzten IP Cores sind in den unten stehenden Kapiteln beschrieben.

JTAG, Timer und System ID
Der JTAG IP Core ermöglicht das flüchtige Programmieren des Nios wie auch das Kommunizieren mit selbem für Debugging Zwecke. Durch den Einsatz des Timer IP Cores erhält der Nios einen Interval Timer um beispielsweise periodisch Interrupts zu generieren. In dem System ID IP Core ist die Systemidentifikationsnummer gespeichert. Diese wird benötigt um beim laden der Software sicherzustellen, dass das passende Hardware Image vorhanden ist. Alle drei Komponenten sind mit Standardeinstellungen in das Nios System eingefügt worden.

Speicher

Der Arbeitsspeicher ist ein externer 64MB SDRAM Chip IS42S16320D von ISSI. Für die Kommunikation mit dem Nios Prozessor ist der SDRAM Controller IP Core zuständig. Der Nios Prozessor kann über das Memory Mapped Interface mit dem Core Kommunizieren und so auf das SDRAM zugreifen. Da dieser Chip auf dem Entwicklungsboard sowieso vorhanden ist, haben wir uns gegen Onchip Speicher entschieden um Ressourcen zu sparen.

Das Hardware Image und der Programmcode ist auf dem Board enthaltenen Flash Speicher gespeichert. Dabei wird anders als bei dem nicht flüchtigen Programmieren nicht das SRAM Object File (.sof) geladen sondern ein JTAG Indirect Configuration File (.jic). Dieses kann in Quartus aus dem SRAM Object File und dem in Eclipse generierten HEX File erstellt werden. Anschliessend kann es über ein Serial Flash Loader Image über den USB Blaster auf den Flash geladen werden. Beim Einschalten des Gerätes wird zuerst das Hardwareimage ins FPGA geladen und anschliessend der Programmcode geholt. Auf Empfehlung von Dokumentationen von Intel haben wir uns dafür entschieden den Programmcode durch einen Bootcopier ins SDRAM zu kopieren. Abbildung 3.2 zeigt das Layout des Flash Speichers nach dem Programmieren. [3]

LCD Controller & Reset

Für das beschreiben des LCD ist die von terasIC bereitgestellte VHDL Komponente LT24_Controller zuständig. Dieser kann über das Memory Mapped Interface von dem Nios Prozessor gesteuert werden. Das verwendete Display LT24 von terasIC enthält für das Schreiben des LCD den LCD Treiber ILI9341 von ILITEK. Dieser Chip wird durch den LT24_Controller über das parallele 16 Bit Interface gesteuert. Weiter kann der LCD Chip über den PIO Core LCD Reset zurückgesetzt werden. Wie diese beiden Komponenten in Software angesteuert werden ist in Kapitel **??** genauer beschrieben. [4]

Touchscreen
Der Touch Screen Digitizer AD7843 von Analog Devices misst den resistiven Touchscreen des LCD aus und übermittelt die digitalisierten Koordinaten über SPI an den Prozessor. Der Nios Kommuniziert dabei über drei verschiedene IP Cores mit diesem Chip. Der SPI Core *Touch SPI*

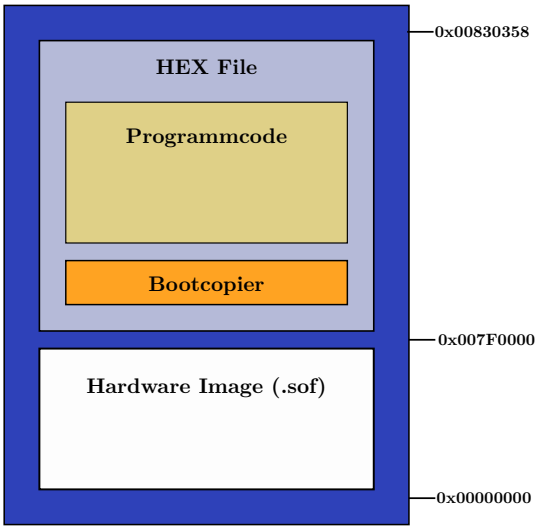


Abbildung 3.2: Layout des Flash Speichers

für die Datenübertragung, der PIO Core *Touch Busy* um den Beschäftigungsstatus des Chips zu wissen und den zweiten PIO Core *Touch Interrupt*, welcher den Nios über eine Betätigung des Touchscreens informert. Bei einer Berührung des Touchscreens löst *Touch Interrupt* beim Nios Prozessor einen Interrupt aus, welcher sofort die Koordinaten über SPI anfordert. [5]

3.4 Pitch Generation

Die Hauptaufgabe der Komponente Pitch Generation ist es das Audiosignal aus dem Rechtecksignal der Tonhöhenantenne (Pitch Antenna) zu generieren. Pitch Generation nimmt zudem eine Frequenzmessung des Audiosignals vor um diverse Funktionalitäten zu gewährleisten. In Abbildung 3.3 ist der Grobe aufbau der Komponente aufgezeigt. Die genaue Erklärung zu den einzelnen Komponenten ist in den folgenden Abschnitten zu finden.

Referenzoszillator

Der Referenzoszillator ist wie in Kapitel ... dafür zuständig ein Sinussignal mit einer Frequenz nahe der des Antennenoszillator zu generieren. Er generiert diesen wie schon erwähnt mithilfe des Cordic Algorithmus. Er ist aufgeteilt in zwei Komponenten: der Cordic Processor und der Cordic Controller. Wie diese beiden Komponenten miteinander verbunden sind ist in Abbildung 3.4 ersichtlich. Beide Komponenten stammen aus dem Projekt 5. Änderungen, welche im Projekt 6 stattfanden, sind entsprechend gekennzeichnet.

Referenz auf Grundlagen digitales Theremin

Der Cordic Processor ist die eigentliche Implementierung des Cordic Algorithmus wie er in Kapitel ... beschrieben wurde. Er muss jedoch für den Einsatz im FPGA leicht angepasst werden. Die Berechnung der Werte für $\arctan 2^{-i}$ konnte vorgängig gemacht werden und ist in eine Lookup Table gespeichert. Dies spart Ressourcen für diese komplizierte Berechnung ein. Wir haben uns entschieden den Algorithmus in einer Pipeline zu implementieren. Dies führt einerseits zu einer höheren maximalen Clockfrequenz, andererseits aber auch zu einer grösseren Signallatenz. Dies ist jedoch nicht problematisch für die gewählte Anwendung, da die Latenz im Nanosekundenbereich ist und später nicht hörbar auffällt. Zuletzt ist eine Multiplikation mit 2^{-i} ganz

Referenz auf Cordic Kapitel einfügen

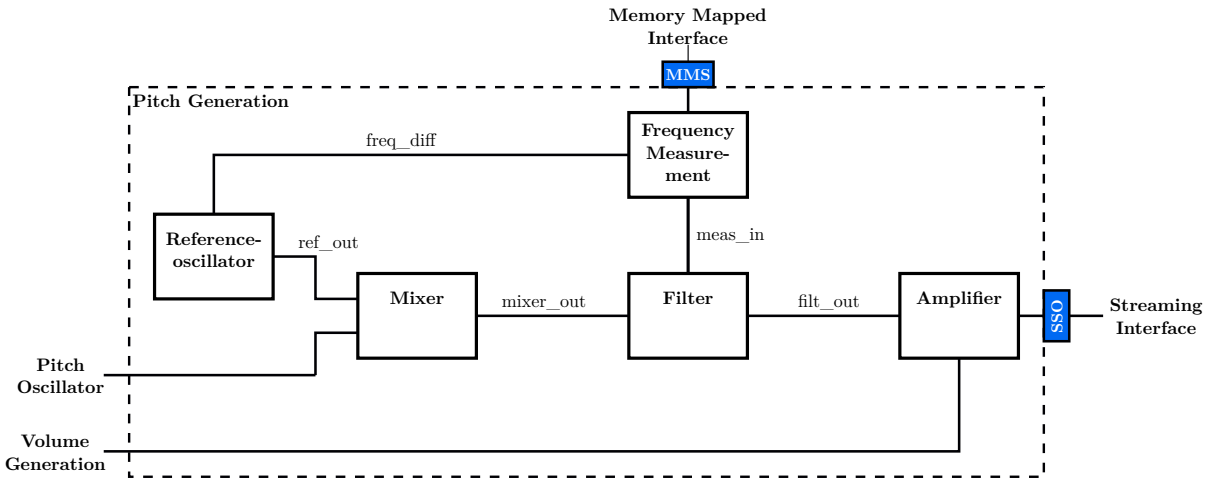


Abbildung 3.3: Blockschaltbild der Custom IP Pitch Generation

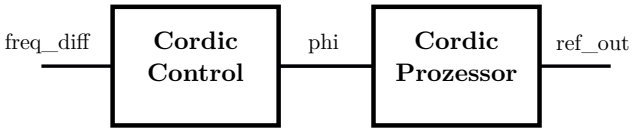


Abbildung 3.4: Aufbau des Referenzoszillators

einfach durch eine Verschiebung um i Bits nach rechts ersetzbar. Dies spart wiederum Ressourcen ein.
Das berechnete Resultat ist als signed Zahl definiert um die Berechnung von negativen Zahlen zu ermöglichen. Sie sind im fixed-point Format und haben 16 Bit Länge. Dabei sind von den 16 Bit 1 Bit Vorzeichen und 15 Bit Nachkommastellen. Dies entspricht einem Zahlenbereich von -1 bis 0.999969.

Für die Berechnung der Sinuswerte muss ein Winkelwert berechnet werden, der mit der Zeit so ändert, dass sich am Ausgang des Cordic Processor ein Sinus mit der gewünschten Frequenz ergibt. Für diese Aufgabe ist der Cordic Controller zuständig. Bei mit der Zeit linear ansteigendem Winkelwert ergibt sich am Ausgang die gewünschte Sinusform. Wichtig ist jedoch, dass der Cordic Algorithmus nur für Winkelwerte zwischen $-\pi/2$ und $\pi/2$ der anders für Werte im ersten und zweiten Quadranten konvergiert. Die Lösung für dieses Problem ist in Abbildung 3.5 ersichtlich. Als erstes wird der Sägezahn Winkel berechnet. Für den linearen Anstieg des Winkels zählt der Cordic Controller einen Zähler mit einer bestimmten Schrittweite jeden Clockzyklus hoch. Der wrap-around des Zählers ist dabei sogar erwünscht um den Sprung zwischen dem II und III Quadranten zu erzielen. Der Schrittwert ergibt sich wie folgt:

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

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 FPGA.

Nun kommt die bereits erwähnte Einschränkung des Cordic Algorithmus ins Spiel. Die berechneten Werte des Sägezahnwinkels zwischen dem II und III Quadranten konvergieren nicht. Aus diesem Grund konvergiert der Controller diesen Winkel in den Dreieckswinkel. Sind die beiden vordersten Bits entweder 01 oder 10 befindet sich der Winkelwert im II respektive III Quadranten. Um in diesem Fall den Dreieckswinkel zu erhalten invertiert der Cordic Controller alle Bits ausser dem MSB. Wie man sich leicht davon überzeugen kann ergibt der Dreieckswinkel denselben Sinusverlauf wie der Sägezahnwinkel bei einer Sinusrechnung ohne die erwähnten Einschränkungen. [6]

Um die Kalibrierung und den Glissandoeffekt für das Theremin zu ermöglichen waren im Projekt 6 kleine Anpassungen am Cordic Controller nötig. Wie zuvor hat der Controller eine fixe Frequenz implementiert, welche in der Grössenordnung 550 kHz liegt. Jedoch legt die Frequenzmessungskomponente nun eine Differenz über einen Eingang an den Cordic Controller an um die zuvor genannten Features über den Referenzoszillator zu ermöglichen.

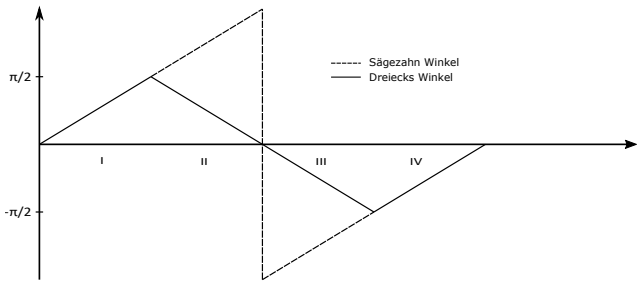


Abbildung 3.5: berechneter Winkel *phi* des Cordic Contrllers in Funktion der Zeit

Mixer

Die Implementation des Mixers ist dank der Entscheidung für die Rechteckform des analogen Oszillatorsignals sehr einfach. Über einen GPIO liest die Mixerkomponente das Signal des Antennenoszillators ein und verrechnet es mit dem generierten Sinus *ref_out*. Eine 1 des Rechtecks wird dabei als die Zahl 1 und eine 0 als die Zahl -1 interpretiert. Die Multiplikation zwischen der 1 und *ref_out* ist dabei nicht nötig und eine Multiplikation mit -1 erzielt der Mixer durch das Bilden des Zweierkomplement von *ref_out*.

Filter

Im Projekt 5 war das Filter ein einzelnes CIC-Filter mit dem Decimationsfaktor 1000. Dies hatte zur Folge, dass viel Aliasing entstand. Um dieses zu verringern haben wir uns im Projekt 6 für den Aufbau aus Abbildung 3.6 entschieden. Die drei Filter CIC 1 bis CIC 3 sind Instanzen einer CIC-Filter Komponente. Diese Komponente ist unverändert geblieben seit Projekt 5. Die Parameter der drei Instanzen sind in Tabelle 3.2 ersichtlich. Da es bei CIC-Filtern wie in Kapitel ... beschrieben, um deren Nullstellen zu Aliasing kommt, sind diese Filter so eingestellt, dass sich die Oberwellen des Rechteck möglichst nicht in deren Nähe befinden. Bei CIC 1 wurde eine höhere Ordnung gewählt um am Anfang eine stärkere Dämpfung zu erzielen. Die Anzahl Ausgangsbits wurde nach Formel ... berechnet.
Zuletzt haben wir noch ein FIR-Filter implementiert um das Signal auf 48kHz unter abzutasten.

Referenz auf CIC-Filter Kapitel einfügen

referenz auf Formel in Theorie einfügen

Das Filter hat eine Passfrequenz von 2 kHz und eine Stopfrequenz von 24 kHz mit einer Dämpfung von 55dB. Wir entschieden uns die Koeffizienten mit dem filterDesigner Tool von Matlab zu berechnen und als 27 Bit signed Zahlen in einer Lookup Table zu speichern. Wir wählten deshalb 27 Bit, da im FPGA eine solche Multiplikation noch knapp in einen DSP Block integriert werden kann. [7] Weswegen der Ausgang Frequency Measurement nach dem zweiten CIC-Filter gewählt wurde ist im nächsten Abschnitt beschrieben.

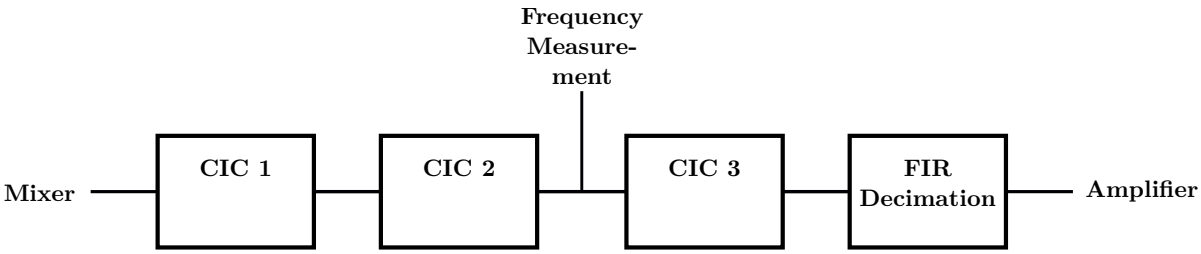


Abbildung 3.6: Aufbau des Filters in der Komponente Pitch Generation

Tabelle 3.2: Parameter der drei CIC-Filter

Komponente	Dezim.Fakt.	Ordnung	Ausgangsfreq.	Ausgangsbits
CIC 1	5	2	10.8 MHz	21 Bits
CIC 2	9	1	12 MHz	25 Bits
CIC 3	5	1	240 kHz	28 Bits

Amplifier

Die Komponente Amplifier ist dafür zuständig beim Signal *filt_out* den Gain der CIC-Filter zu kompensieren und anschliessend dieses mit der Dämpfung, welche die Volume Generation liefert, zu multiplizieren. In Abbildung 3.7 ist eine Problematik aufgezeigt, welche auftritt, wenn man die Dämpfung zu beliebigen Zeiten wechselt. Links sieht man, was passiert, wenn die Dämpfung beim höchsten Wert der positiven Halbwelle ändert. Diese Sprünge im Signal treten dann als hörbares Knacksen auf. Um dies zu verhindern ist eine Erkennung von Nulldurchgängen implementiert, dass wie in der Abbildung rechts die Dämpfungen nur bei diesen Nullstellen geändert werden. Da der Codec ein Offsetbehaftetes Signal verlangt muss das MSB des Signal getoggelt werden um dies zu bewerkstelligen. Diese Komponente enthält zudem die Kommunikation mit dem Streaming Interface.
Die Dämpfung des Audiosignals könnte auch über den Codec gemacht werden. Weshalb dies nicht möglich ist, ist in Kapitel ... beschrieben.

Referenz auf Software Audio

Frequenzmessung, Kalibration & Glissandoeffekt

Die Komponente Frequency Measurement hat mehrere Aufgaben und ist diejenige Komponente mit welcher über den Nios Prozessor die gesammte Pitch Generation gesteuert werden kann. Der Aufbau dieser Komponente ist in Abbildung **??** aufgezeigt.
Zum einen wird hier die Frequenzmessung durchgeführt. Dies geschieht über die drei Komponenten FIR, Period Counter und Goldschmidt divider. FIR ist wie der Name sagt ein FIR Filter. Dieses ist nötig um das Signal aus dem Filter, welches noch Hochfrequente Signale enthält zu Filtern. Das Filter hat eine Grenzfrequenz von ... . Das Filter wurde mit dem Tool filterDesigner

Grenzfrequenz und Dämpfung einfügen

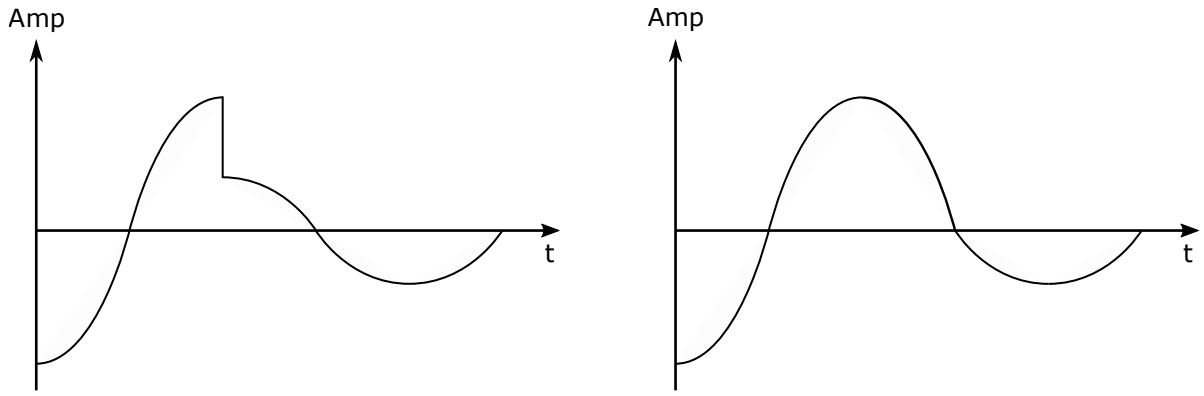


Abbildung 3.7: Unterschied Dämpfungswechsel (links ohne und rechts mit Nullstellenerkennung)

in Matlab berechnet. Wir haben entschieden die Filterkoeffizienten als fixed-point signed Zahlen in einem Array mit 18 bit länge abzuspeichern. Auf dieser Koeffizientenlänge kann Quartus die DSP-Blöcke so nutzen, dass nach der Multiplikation des Signals mit den Koeffizienten die Resultate gleich in den Blöcken addiert wird. Dies ermöglicht längereMultiplikationsketten. Anschliessend wird das Signal *fir_out* im *Period Counter ausgemessen*. Diese zählt von Nulldurchgang zu Nulldurchgang einen Zähler hoch. Bei einem Nulldurchgang wird der Wert dieses Zählers am Signal *per_cnt* ausgegeben. Der Zählerwert entspricht der Anzahl Abtastwerte des Signals in einer Signalperiode.
Das Signal hat eine Abtastfrequenz von 1.2 MHz. Dividiert man diese Abtastfrequenz durch die zuvor gezählte Anzahl Abtastperioden erhält man die Frequenz des Signals.

Nochmal über DSP blöcke nachlesen und referenzieren

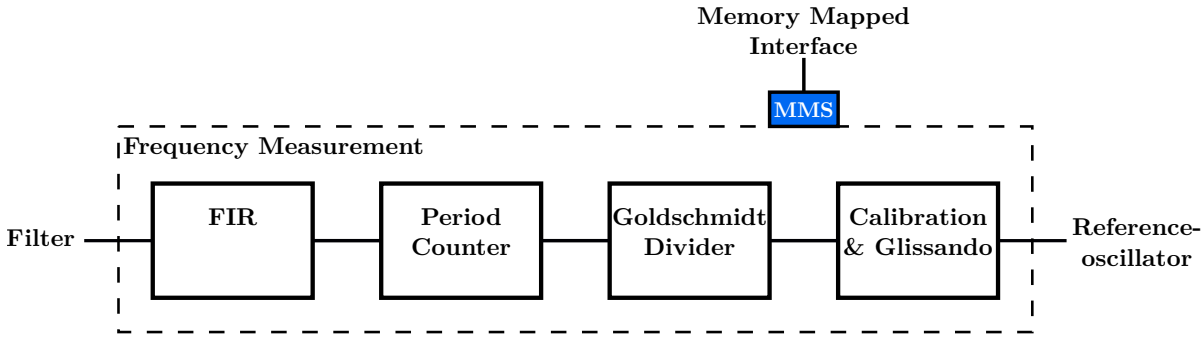


Abbildung 3.8: Aufbau der Frequenzmessung, Kalibration und Glissandoeffekt in der Komponente Pitch Generation

Bild überarbeiten und namen an Signalen anbringen wie in Text

3.5 Volume Generation

Das digitale Theremin ist auf dem Entwicklungsboard DE1-SOC von terasIC aufgebaut. Dieses enthält ein Cyclone V 5CSEMA5 FPGA von Intel. Weiter befindet sich auf dem Board der Audio Codec WM8731 von Wolfson für die Ausgabe an einem Lautsprecher. In Abbildung ... ist der Aufbau des Digitalen Theremin aufgezeigt inklusive der Peripherie ausserhalb des FPGA.

Referenz auf Blockschaltbild

Filter

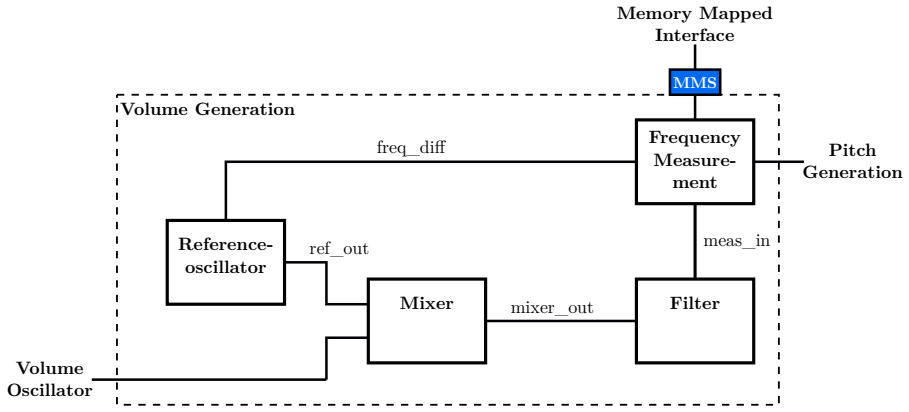


Abbildung 3.9: Blockschaltbild der Custom IP Volume Generation

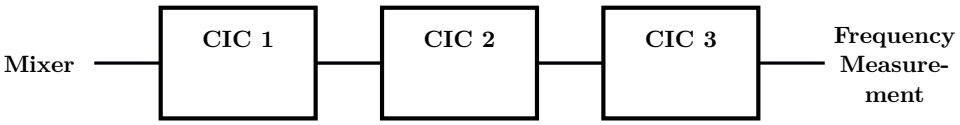


Abbildung 3.10: Aufbau des Filters in der Komponente Volume Generation

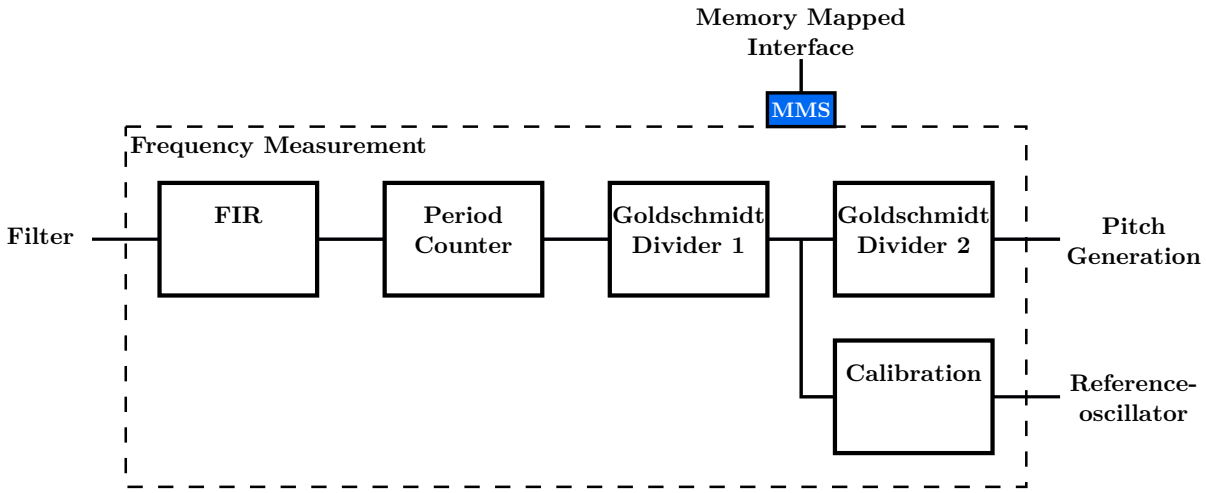


Abbildung 3.11: Aufbau der Frequenzmessung und Kalibration in der Komponente Volume Generation

Frequenzmessung & Kalibration

4 Realisierung Software

Im folgenden Teil des Fachberichtes ist der Aufbau der Software beschrieben. Es folgt zu Beginn eine Gesamtübersicht.

Die Software ermöglicht die Bedienung des Theremin. Zur Steuerung des Theremin ist auf dem NIOS II eine in C programmierten Statemachine realisiert. Die Bedienung geschieht über das LT24 LCD touch module von Terasic. Die von Terasic zur Verfügung gestellten C Dateien zur Ansteuerung des LCD und des Touch, sowie die Dateien zur Darstellung des GUI, stellten sich als wenig brauchbar heraus. Die darin enthaltenen Funktionen sind oft zu ineffizient. Daher entschieden wir uns diese Funktionen selber zu schreiben.

4.1 Hauptprogramm State Machine

Abbildung 4.1 zeigt die Initialisierung und den Hauptprogrammfluss der Software.

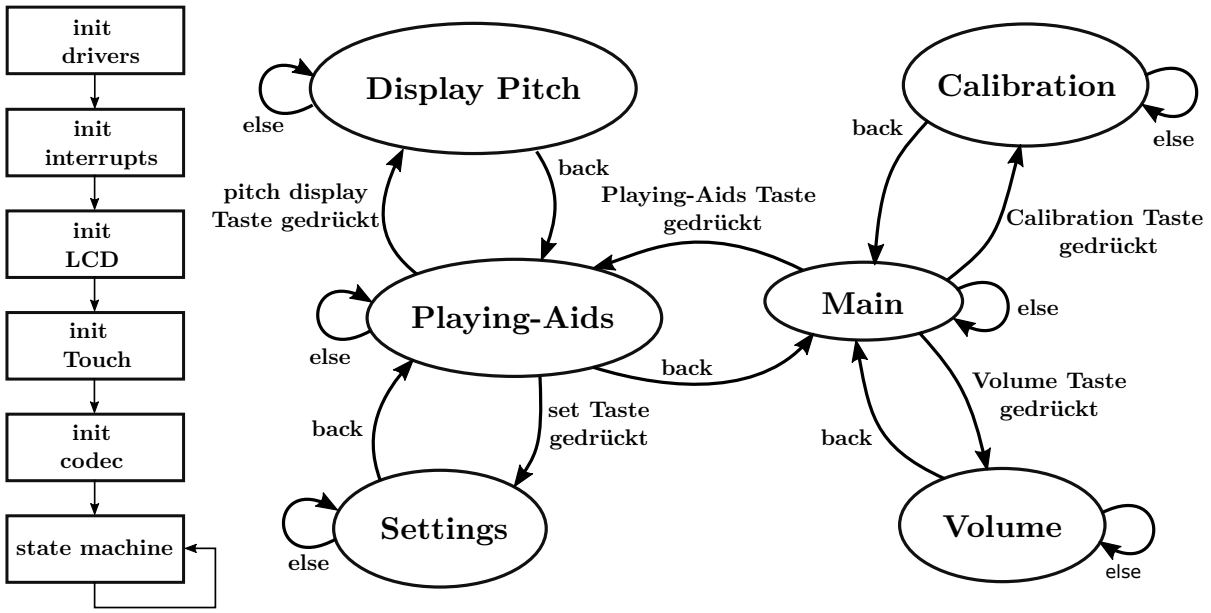


Abbildung 4.1: State Machine und Initailisierung.

Als erstes initialisiert das Programm alle Treiber, der Touch, das LCD, und der Codec. Anschlissend führt das Programm in einer Endlosschlaufe die state machine durch.

Main: Dies ist der erste State nach dem Initialisierungsvorgang. Der Main State wartet auf eine Betätigung der drei in Abbildung 4.2a gezeigten Tastern. Eine Betätigung der drei Tastern führt zu einem Wechsel in den entsprechenden State.

Calibration: Der Calibration State fordert den Benutzer auf seine Hände in die Nähe der Antenne zu halten. Nach zwei Sekunden startet die Kalibrierung. Sobald die Kalibrierung abgeschlossen ist kann der Benutzer über den ***back*** Taster zurück in den Main State gelangen. Abbildung 4.2b zeigt die Darstellung des LCD nach erfolgreicher Kalibrierung.

Volume: In diesem State kann der Benutzer die Lautstärke ändern und die Volume Antenne aktivieren und deaktivieren. Die Lautstärke kann in 10 verschiedene Pegeln eingestellt werden. Dies geschieht mithilfe der in Abbildung 4.2c gezeigten + und - Tasten. Beim betätigen der ***vol***

antenna Taste wird die Volume Antenne je nach aktuellem Zustand deaktiviert oder aktiviert. Mit dem betätigen der ***back*** Taste gelangt der Benutzer in den Main State.

Playing-Aids: Im State Playing Aids kann der Benutzer mit der ***Glissando on*** Taste der Glissando Effekt akieviert und deaktiviert. Die Abbildungen 4.2d und 4.2e zeigen die grafische Realisierung dieser Taste. Über die ***Set*** Taste gelangt der Benutzer in den State Settings. Durch betätigen des ***display pitch*** Taste wird in den State Pitch Display gewechselt. Zurück in den Main State gelangt man über die ***back*** Taste.

Settings: Im Settings State können Einstellungen zum Glissando Effekt gemacht werden. Das Delay des Glissando Effekts ist in 10 Stufen einstellbar. Zudem kann der Anwender mit dem Taster ***penta*** zwischen der Pentatonischer und der normalen Tonleiter wechseln. Mit dem betätigen der ***back*** Taste gelangt der Benutzer in den Playing-Aids State. Abbildung 4.2f zeigt wie der Setting State grafisch auf dem LCD aussieht.

Pitch Display: Dieser State unterstützt den Theremin Spieler dabei Töne der Tonleiter zu treffen. Abbildung 4.2g zeigt die graphische Darstellung dieses States. Der Schriftzug oberhalb des LCD zeigt den Ton an, der sich in der Region der gespielten Frequenz befindet. Der kleine vertikale Strich zeigt dem Spieler graphisch an wie weit weg die aktuell gespielte Frequenz von dem Ton der Tonleiter ist. Diese graphische Untertützung kann jedoch nur bei der Pentatonischen Tonleiter angewendet werden. Da die Antenne sehr empfindlich auf Änderungen ist, ist es mit der normalen Tonleiter nicht möglich den Vertikalen Strich anzuzeigen. Daher wird dieses Anzeige nur bei der Pentatonischen Tonleiter verwendet.

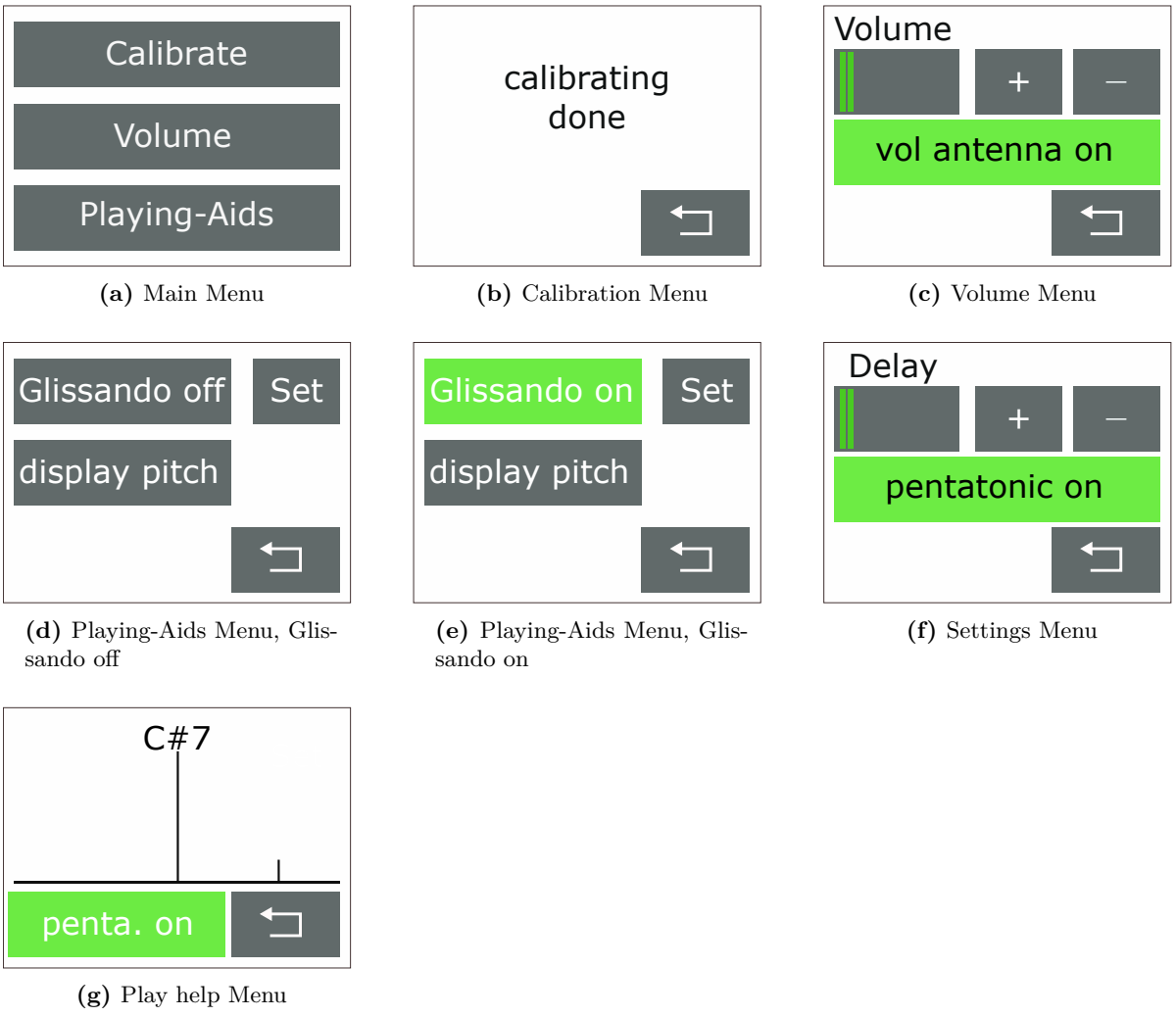
4.2 Treiber

In diesem Unterkapitel sind die verschiedenen Funktionen der selbst geschriebenen Treiber für die Custom IP componenten Pitch generation und Volume generation beschrieben. Zusätzlich war die Erstellung eines Treiber für den LT24 Controller nötig, da Terasic den zur Verfügung gestellte Treiber nicht nach dem Konventionen von Intel erstellt hat.

Um ein selbst erstellten Treiber dem Board Support Package (BSP) hinzuzufügen, müssen die Benennungen und Ablage Orte der Files einige Bedingungen einhalten. Die Treiber Dateien müssen in einem Ordner IP abgelegt sein, welcher sich im quartus Ordner befinden muss. Darin muss ein Skript mit der Endung sw.tcl abgelegt sein. In diesem Skript muss ein eindeutiger Name für den Treiber angegeben sein. Zudem muss der Pfad zu den Treiber Daten angegeben sein. Intel empfiehlt drei Treiber Daten zu erstellen:

- inc
  - custom_ip_regs.h
- HAL
  - custom_ip_.h
  - custom_ip_.c

Das file mit der Endung regs.h definiert hardware interface spezifische Abläufe. Dieses wird im Ordner inc abgelegt. Im HAL Ordner sind ein c und ein h file erstellt, welche die Integration mit dem hardware abstracten layer(HAL) ermöglichen [8]. Die folgenden Paragraphe zeigen die in den c Dateien realisierten Funktionen für die drei Treiber.



(a) Main Menu

(b) Calibration Menu

(c) Volume Menu

(d) Playing-Aids Menu, Glissando off

(e) Playing-Aids Menu, Glissando on

(f) Settings Menu

(g) Play help Menu

Abbildung 4.2: Dummy figure

LT24 Controller Wie in der Einleitung dieses Unterkapitels erwähnt erfüllt der von Terasic mitgelieferte Treiber nicht die Konventionen welche Intel verlangt. Zudem sind die meisten Funktionen des Treibers sehr ineffizient gestaltet. Darum haben wir uns entschieden den Treiber für den LT24 Controller selbst zu schreiben. Im folgenden Teil ist beschrieben welche Funktionen wir für die Steuerung des LCD erstellt haben.

Das Mudul **LT24_Controller.c** ermöglicht es auf dem LCD einzelne Pixel und Rechteckflächen zu zeichnen und zu löschen. Die Funktion *LCD_DrawPoint(x,y,color)* setzt ein Cursor an die gewünschte Stelle auf dem LCD und zeichnet ein Pixel in der entsprechenden Farbe. Auf diese Art ein Rechteck zu zeichnen ist sehr ineffizient. Da der Treiber von Terasic Rechtecke auf diese Art zeichnet, haben wir entschieden eine bessere Lösung zu finden. Mit der Funktion *LCD_DrawRect(xs,ys,xe,ye,color)* werden Rechtecke effizienter gezeichnet. Es wird dabei nur einmal für das ganze Rechteck ein Cursor Feld gesetzt und danach alle Pixel eingefärbt. Da durch das setzen des Cursor Feld nicht für jeden einzelnen Pixel ein Cursor gesetzt werden muss, wird Zeit eingespart. Dadurch ist es für das Menschliche Auge nicht mehr ersichtlich wie einzelne Pixel gezeichnet werden, wie dies bei der Funktion von Terasic der Fall ist.

Pitch generation Das Modul **Pitch_generation.c** ermöglicht auf das Kontroll und das Glissando Delay Register zu schreiben und diese zu lesen. Das Register freq data ist nur lesbar. Dieses liefert die aktuelle Frequenz der Pitch Antenne und der Index des Tones bei dem die Frequenz am nächsten liegt. Die Funktion *get_pixel_pitch_accuracy(penta_on_off,pitch_freq)* liesst das freq data Register aus. Für den Modus Pitch Display, zu sehen in Abbildung 4.2g, ist es nötig anzuzeigen auf welcher Seite sich der Benutzer von einem Ton befindet. Daher wird in dieser Funktion das Verhältnis der aktuellen Pitch Antenna Frequenz mit dem Abstand zum nächsten Ton gebildet. Mit diesem Verhältnis wird ein Pixel Wert berechnet. Dieser wird gebraucht um denn vertikalen Strich auf dem LCD zu zeichnen.

Volume generation Die einzige Kominikation welche die Volume generation Komponente mit dem NIOS II hat, ist das schreiben und lesen des Kontroll Registers um die Kalibrierung zu starten. Das Module *Volume_generation.c* ermöglicht dies.

4.3 Audio

Die Kommunikation mit WM8731 Codec geschieht über die IP Componente Audi_Video Configuration von Intel. Diese besitzt die Funktion *alt_up_av_config_write_audio_cfg_register* um die Kontrollregister des Codec über den Hardware Abstraction Layer (HAL) anzusprechen. . Das Module **audio.c** beinhaltet die beiden Funktionen, *codec_wm8731_init()* und *set_vol(vol_gain)*.

Quelle Intel

In der Initialisierung wird der Codec als Master konfiguriert . Der Clock des Codec wird auf 12 MHz gesetzt und die Sampling Rate auf 48 kHz. Die Input Audio Data Bit Länge wird auf 24 Bit gesetzt und der Übertragungsmodus der Daten auf left justified gesetzt. Mit diesen Einstellungen ist eine Kominikation mit der audio serializer Komponente möglich . Um Störungen zu vermeiden sind die Line In Eingänge des Codec gemutet, da wir nur den Line Out brauchen. Der Linke und Rechte Kanal sind so eingestellt das beide Kanäle die selben Lautstärken haben[9].

Verweiss auf Dennis

Verweiss auf Dennis

Die Gesamtlautstärke des Theremin ist auf 10 unterschiedliche Pegel einbestellbar, dies geschieht mit dem Aufruf der Funktion *set_vol(vol_gain)*. Die leiseste Stufe dämpft den Audio Signal Pegel um −76 db. Jeweils eine Stufe grösser reduziert die Dämpfung um 7 db. Die höchste Stufe dämpft den Audio Signal Pegel um −7 db. Der Codec könnte gemäss Datenblatt das Signal bis +6dB verstärken. Nach Labor Tests haben wir uns entschieden das Audio Signal nur zu dämpfen, da eine Verstärkung zu laut ist.

Ursprünglich war geplant die Lautstärke Einstellung der Volume Antenne über den Codec zu steuern. Nicht wie in Kapitel beschrieben über die VHDL Komponente. Diese Methode konnte jedoch nicht realisier werden, da die Zero Cross detection des Codec nicht funktionierte. Mit der Zero Cross Dedection sollte der Codec erst bei einem Nulldurchgang die Lautstärke vermindern oder erhöhen. In unseren Versuchen hat sich die Lautstärke jedoch nie geändert, obwohl ein Sinus Audio Signal angelegt wurde. Wir mussten feststellen das diese Methode für den DAC nicht funktioniert. Mit dieser Methode hätte die Lautstärke, verglichen zur realisierten Methode, eine bessere Dynamik gehabt.

Dennis

4.4 Touch

Die von Terasic zur Verfügung gestellte Datei zur auslesen des Touch ist leider sehr unübersichtlich aufgebaut. Es ist sehr schwierig die darin enthaltenen Funktionen auf unser Projekt anzuwenden. Daher entschieden wir uns selbst ein Touch Interrupt Routine zu erstellen.

Der resistive Touch Display des LT24 LCD touch module ist mit dem AD783 Analoge Devices Chip verbunden. Sobald der Touch Display berührt wird lösst der Chip am Pen Interrupt Pin ein Interrupt aus welches vom NIOS II detektiert wird. Der AD783 Chip speicher die X und Y Koordinaten zum Zeitpunkt des Interrupts ab. Diese können über den SPI Bus ausgelesen werden[10].

Quelle Datenblatt AD7843

Für unser Projekt sind wir daran interessiert den Pen Interrupt zu detektieren und die X und Y Koordinaten auszulesen, damit wir sagen können welcher Taster gedrückt worden ist.

Das ganze ist im Modul **touch_isr.c** realisiert. Darin befindet sich eine Funktion *touch_init(void*context)*. Diese aktiviert das Touch Pen Interrupt und registriert die Funktion welche durch das Interrupt aufgerufen wird. In der Interrupt Funktion *touch_isr(void*context)* wird zuerst das Touch Pen Interrupt deaktiviert. So das in dieser Zeit kein weiteres Interrupt auftreten kann. Nach dem deaktivieren des Interrupts liesst der *alt_avalon_spi_command* die X und Y Koordinaten aus.

4.5 GUI

Die Funktionen welche von Terasic für die Gestaltung des GUI zur Verfügung gestellt werden sind für unser Projekt zu ineffizient. Um ein Text anzuzeigen verwendet Terasic eine Funktion die ein Alpha Blending durchführt. Dieses wird immer an den Rändern der Buchstaben durchgeführt. Dabei wird die Schriftfarbe mit der Hintergrundfarbe gemischt. Dieser Prozess nimmt viel Zeit in Anspruch. Wodurch bei jedem neuen zeichnen eines Texte Strings zugeschaut werden kann wie die Pixel gezeichnet werden. Dies ist für unser Projekt unbrauchbar. Zudem hat uns die von Terasic verwendete Schriftart nicht gefallen. Daher entschieden wir uns eine eigene Funktion zu schreiben welche Text stringe zeichnet. Diese ist im Modul **simple_text.c** realisiert.

simple_text.c Da das LCD nur in der Lage ist Pixel zu setzen gestaltet sich das zeichnen einer Schrift eher mühsam. Mit dem open source tool "The Dot Factory"generierten wir uns eine Bitmap für die Schrifftart Arial welche 22points gross gewählt ist. Das tool generiert zwei Arrays. Im Character Bitmap Array sind die Character in einer Bitmap gespeichert. Das Character Descriptor Array enthält Informationen über die Breite jedes Characters und den Offset in der Character Bitmap. Um den Ort eines Characters zu bestimmen wird der Character minus des ersten Character in der Bitmap gerechnet. Dies ergibt den Index für das Descriptor Array in welchem der Offset für die Bitmap gespeichert ist. In Abbildung 4.3 ist dieser Prozess veranschaulicht. In diesem Beispiel besteht die Bitmap aus den Buchstaben abc. Für den Buchstaben b wird der Offset 1 ausgerechnet. Im Descriptor Arrey ist auf Position 1 die Breite 5 gespeichert und der Offset 10 für die Bitmap. Mit diesen Informationen kann die Funktion *print_string(x,y,color,font,font_descriptor,string)* einen Text String zeichnen. Dabei werden nur die Pixel gezeichnet welche in der Bitmap auf 1 gesetzt sind.

gui.c In diesem Modul sind mit der Funktion *print_string(x,y,color,font,font_descriptor,string)* aus dem Modul **simple_text.c** und der Funktion *LCD_DrawRect(xs,ys,xe,ye,color)* aus dem Modul **LT24_controller.c** die einzelnen Menüs dargestellt.

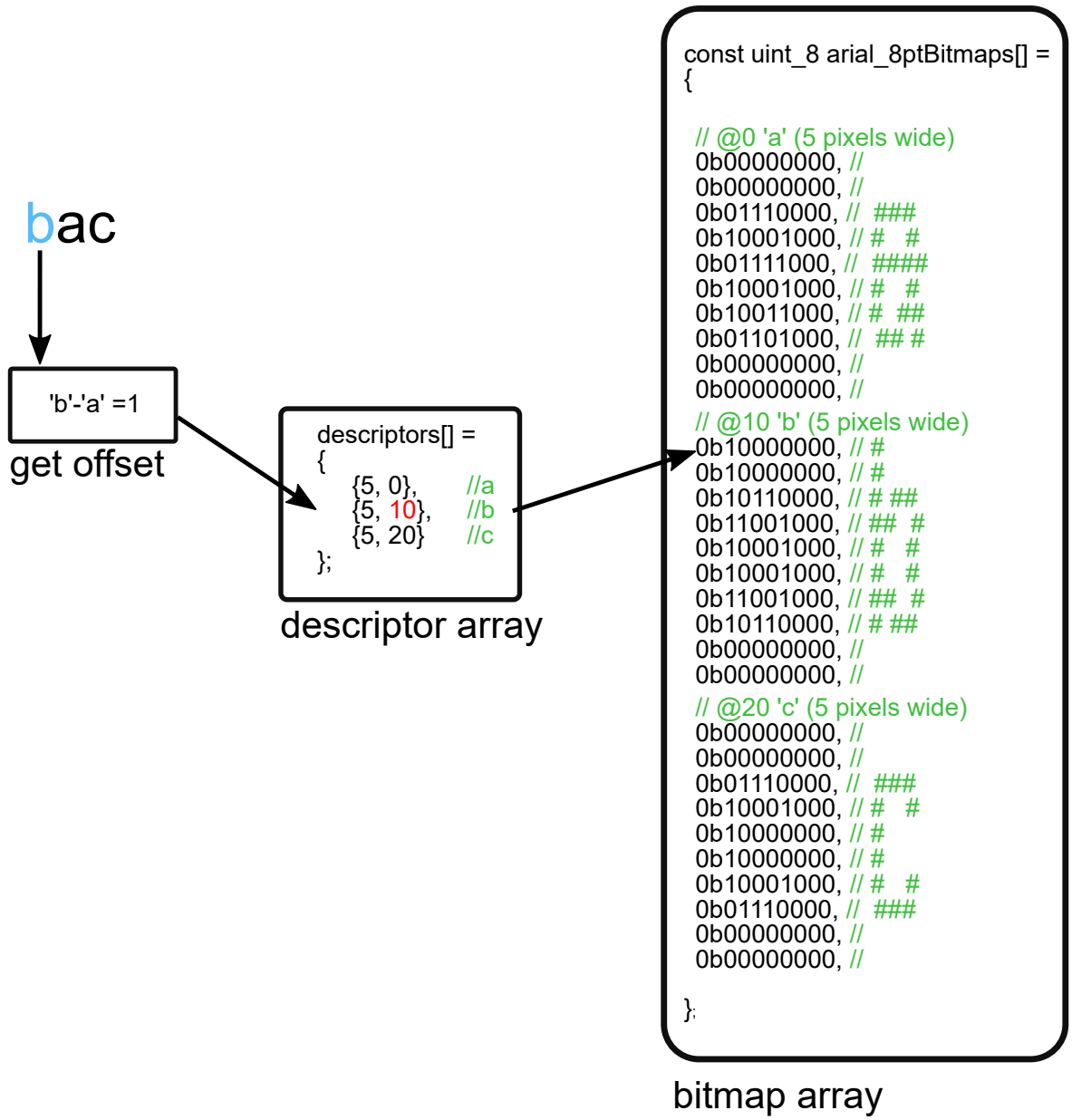


Abbildung 4.3: Bitmap und Discriptor Array.

5 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.

5.1 PCB

bla bla

5.2 Frequenzmessung

bla bla

5.3 Glissando Effekt

bla bla

5.4 Ton Display

bla bla

6 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 Tonhöhenoszillator 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 einstellen 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 Antennenoszillator 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.

7 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] Intel. (2020). Embedded Design Handbook. (Abrufdatum 12.08.2020), Adresse: `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/edh_ed_handbook.pdf`.

[4] ILITEK. (2010). TFT LCD Single Chip Driver ILI9341. (Abrufdatum 12.08.2020), Adresse: `https://cdn-shop.adafruit.com/datasheets/ILI9341.pdf`.

[5] Analog Devices. (2004). Touch Screen Digitizer AD7843. (Abrufdatum 12.08.2020), Adresse: `https://www.analog.com/media/en/technical-documentation/data-sheets/AD7843.pdf`.

[6] Esteban O. Garcia. (2006). Paper Cordic Generator. (Abrufdatum 17.01.2020), Adresse: `https://ccc.inaoep.mx/~rcumplido/papers/2006-Garcia-Pipelined%20CORDIC%20Design.pdf`.

[7] Intel. (2020). Cyclone V Handbook. (Abrufdatum 14.08.2020), Adresse: `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-v/cv_5v2.pdf`.

[8] ——, (2020). Nios® II Software Developer Handbook. (Abrufdatum 14.08.2020), Adresse: `https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/nios2/n2sw_nii5v2gen2.pdf`.

[9] Wolfson. (2012). Portable Internet Audio CODEC WM8731. (Abrufdatum 17.01.2020), Adresse: `https://statics.cirrus.com/pubs/proDatasheet/WM8731_v4.9.pdf`.

[10] Analog Devices. (2004). Touch Screen Digitizer. (Abrufdatum 14.08.2020), Adresse: `https://www.analog.com/media/en/technical-documentation/data-sheets/AD7843.pdf`.