**Zürcher Hochschule für Angewandte Wissenschaften**

Department School of Engineering

Technische Projektdokumentation

# FPGA Audio-Synthesizer

*Autoren:*
Dylan Baumann
Lorena Mastroberti
Madjri Sodzi
Luka Vladetic

*Dozierende:*
Matthias Rosenthal
Rifat Gürboy

Abgegeben am
29. Oktober 2024

Modul:
t.BA.ET.PM2.19HS

# Imprint

*Projekt:* Technische Projektdokumentation
*Titel*: FPGA Audio-Synthesizer
*Autoren*: Dylan Baumann
Lorena Mastroberti
Madjri Sodzi
Luka Vladetic
*Datum*: 29. Oktober 2024
*Keywords*: FPGA, Synthesizer, Digitaltechnik

*Hochschule:*
ZHAW Zürcher Hochschule für Angewandte Wissenschaften

*Modul:*
t.BA.ET.PM2.19HS

*Dozierender 1:*
Matthias Rosenthal
Zürcher Hochschule für Angewandte Wissenschaften
Email: matthias.rosenthal@zhaw.ch

*Dozierender 2:*
Rifat Gürboy
Zürcher Hochschule für Angewandte Wissenschaften
Email: rifat.guerboy@zhaw.ch

# Zusammenfassung

Das Projekt "Digitaler Audio-Synthesizer" wurde im Rahmen des Digitaltechnik-Projektmoduls an der ZHAW School of Engineering durchgeführt, mit dem Ziel, einen digitalen Audio-Synthesizer mithilfe des Intel® MAX® 10 FPGA zu entwickeln. Das Projekt ist in fünf Kapitel unterteilt, die die vier vorgegebenen Meilensteine und die erweiterten Funktionen abdecken. Die Hauptfunktionen umfassen die Implementierung von Direct Digital Synthesis (DDS) zur Erzeugung von Audiosignalen, einen MIDI-Decoder zur Verarbeitung von MIDI-Eingangssignalen, einen Codec-Controller zur Steuerung des Audiocodecs und einen I2S-Master für die digitale Audioübertragung. Zusätzlich wurden erweiterte Funktionen wie eine Bluetooth-App, ein VU-Meter, die FM-Synthese und eine Siebensegmentanimation entwickelt und integriert. Diese Komponenten arbeiten zusammen, um MIDI-Signale zu empfangen, zu verarbeiten und in analoge Audiosignale umzuwandeln. Der Audio-Synthesizer kann sowohl über ein Keyboard als auch über eine App gesteuert werden. Die Ergebnisse zeigen, dass alle gesetzten Ziele erreicht wurden und der Synthesizer erfolgreich alle vorgesehenen Funktionen ausführt.

# Inhaltsverzeichnis

# 1 Einleitung

Das Projekt "Digitaler Audio-Synthesizer" wurde im Rahmen des Digitaltechnik-Projektmoduls an der ZHAW School of Engineering durchgeführt. Der Fokus des Projekts liegt auf der Entwicklung eines digitalen Audio-Synthesizers, der mithilfe des Intel® MAX® 10 FPGA (Field Programmable Gate Arrays) realisiert wird.

Das Projekt ist in fünf Kapitel gegliedert, die die vier vorgegebenen Meilensteine und die eigenständig entwickelten erweiterten Funktionen abdecken. Die vier Meilensteine konzentrieren sich auf die Implementierung wesentlicher Funktionen des Synthesizers: Direct Digital Synthesis (DDS) zur Erzeugung von Audiosignalen, einen MIDI-Decoder zur Verarbeitung von MIDI-Eingangssignalen, einen Codec-Controller zur Steuerung des Audiocodecs und einen I2S-Master für die digitale Audioübertragung. Diese Komponenten bilden die Grundlage für die Funktionalität des Synthesizers und stellen sicher, dass er MIDI-Signale korrekt empfangen, verarbeiten und in analoge Audiosignale umwandeln kann.

Über die vorgegebenen Meilensteine hinaus wurden erweiterte Funktionen in das Projekt integriert. Diese umfassen eine Bluetooth-App zur drahtlosen Steuerung, ein VU-Meter zur Visualisierung der Lautstärkepegel, die FM-Synthese zur Erzeugung komplexerer Klangstrukturen und eine Siebensegmentanimation zur visuellen Darstellung der Audiowiedergabe.

Die folgenden Kapitel dieser Dokumentation werden detailliert auf die Implementierung der einzelnen Komponenten eingehen und die erzielten Ergebnisse präsentieren.

# 2 Projektübersicht des Audio-Synthesizers

Der Audio-Synthesizer setzt sich aus mehreren wesentlichen Komponenten zusammen, die nahtlos zusammenarbeiten, um MIDI-Signale zu empfangen, zu verarbeiten und in analoge Audiosignale umzuwandeln. Dieses Kapitel beschreibt die Interaktion der Ein- und Ausgänge des Audio-Synthesizers sowie die grobe Verarbeitung der MIDI-Daten. Die folgende Abbildung 2.1 stellt den Audio-Synthesizer dar.

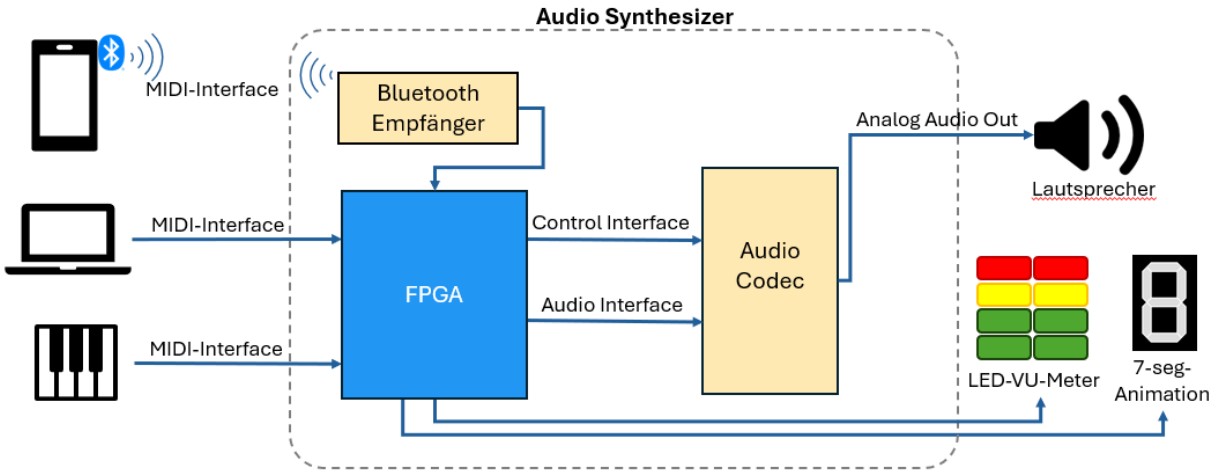


Abbildung 2.1: Systemübersicht

## 2.1 Bluetooth Empfänger

Der Bluetooth-Empfänger, der auf einem von der ZHAW entwickelten Board integriert ist, empfängt die drahtlos übertragenen MIDI-Daten und leitet sie an das FPGA weiter. Dadurch wird das FPGA mittels einer App angesteuert und ersetzt das Keyboard.

## 2.2 FPGA MAX10

Das FPGA MAX10 ist das zentrale Element des Systems. Das FPGA verarbeitet die empfangenen MIDI-Daten und wandelt sie in digitale Audiosignale um. Diese werden über eine Audio-Schnittstelle an den Audio-Codec weitergeleitet. Zudem steuert das FPGA den Audio-Codec an und liefert Daten an die Ausgänge. Eine detaillierte Beschreibung der grundlegenden Signalverarbeitung sind in Kapitel 3 beschrieben. Zusätzliche Beschreibungen zur Signalverarbeitung, die nicht zur Grundfunktion des Audio-Synthesizers gehören, sind in Kapitel 4 beschrieben.

## 2.3 Audio Codec WM8731

Der Audio-Codec WM8731, der ebenfalls auf einem von der ZHAW entwickelten Board integriert ist, wandelt die digitalen Audiosignale vom FPGA in analoge Audiosignale um. Diese analogen Signale werden über eine analoge Audioausgangsschnittstelle an den Lautsprecher weitergeleitet.

## 2.4 VU-Meter

Die Signale werden direkt vom FPGA abgegriffen und an das VU-Meter weitergeleitet. Das VU-Meter ist ein zusätzliches Board, das an das FPGA angeschlossen wird. Es filtert die Signale so, dass die LEDs die Lautstärke darstellen. Die LEDs sind so angeordnet, dass sie einen Pegel anzeigen, der von unten (leise) nach oben (laut) verläuft.

## 2.5 Siebensegmentanimation

Die Animation umfasst sechs Siebensegmente und eine LED-Reihe mit zehn LEDs. Die Animation wird vom Audio-Codec gesteuert. Wenn ein Ton über das Keyboard oder Bluetooth abgespielt wird, stellen die Segmente und LEDs eine Animation zur Rückgabe des Tons dar. Die Segmente stellen ein Muster dar, während die LED-Reihe bei der untersten LED beginnt und bis zur obersten LED durchgehend leuchtet. Dieser Vorgang wird in einer Schleife wiederholt.

## 2.6 Bedienung des System

Der gesamte Audio-Synthesizer kann mit dem Keyboard oder einer App gespielt werden. Werden die Tasten bedient, wird die Lautstärke auf dem VU-Meter visualisiert und die Tonwiedergabe wird mit der Siebensegmentanimation präsentiert.

# 3 Systemarchitektur und Signalfluss der Grundfunktionen auf dem FPGA

Das dargestellte System in Abbildung 2.1 verwendet das FPGA MAX10 als zentrale Steuereinheit und integriert mehrere wichtige Komponenten, um MIDI-Daten in analoge Audiosignale umzuwandeln. Im Folgenden werden die einzelnen Komponenten und ihre Funktionen im Detail beschrieben. Über die Funktionsweise des Audio Codec WM8731 wird in diesem Kapitel nicht eingegangen, da sich dieser auf einem externen Board befindet.

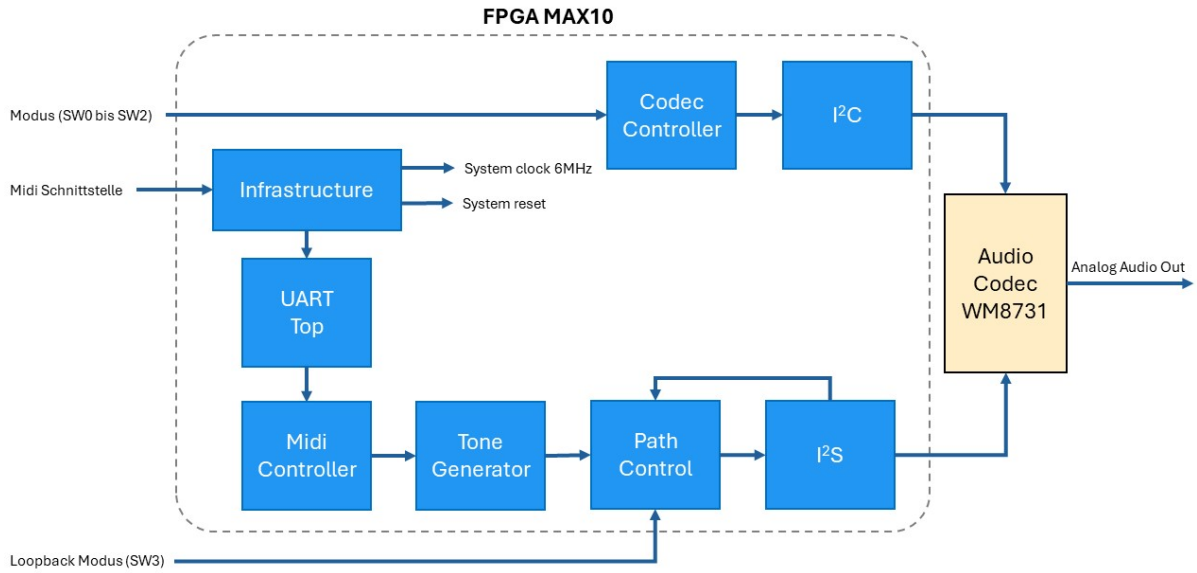


Abbildung 3.1: Systemarchitektur der Grundfunktionen

## 3.1 Infrastruktur

Die Infrastruktur-Komponente stellt grundlegende Systemressourcen bereit, die für den Betrieb des gesamten Systems notwendig sind. Dies umfasst den Systemtakt, der mit einer Frequenz von 6 MHz betrieben wird, und das System-Reset-Signal. Der Systemtakt synchronisiert die Operationen aller digitalen Komponenten im FPGA, während das Reset-Signal das System in einen definierten Startzustand zurücksetzt. Zusätzlich synchronisiert die Infrastruktur das Eingangssignal, um sicherzustellen, dass alle eingehenden MIDI-Daten zeitlich korrekt verarbeitet werden.

## 3.2 UART (Universal Asynchronous Receiver-Transmitter)

UART Top ist für die serielle Kommunikation innerhalb des Systems verantwortlich. Dieses Kommunikationsprotokoll ermöglicht die asynchrone Datenübertragung. Der UART-Controller empfängt serielle MIDI-Daten und wandelt sie in parallele Daten um, die dann vom restlichen System verarbeitet werden können. Es dient als Schnittstelle zwischen der MIDI-Schnittstelle und den internen Verarbeitungsmodulen des Systems.

## 3.3 MIDI-Controller und Tone Generator mit DDS

Das System zur Klangerzeugung ist modular aufgebaut und nutzt Direct Digital Synthesis (DDS), um Klänge zu erzeugen. Wie in Abbildung 3.2 dargestellt, steuert der MIDI-Controller die DDS-Module.

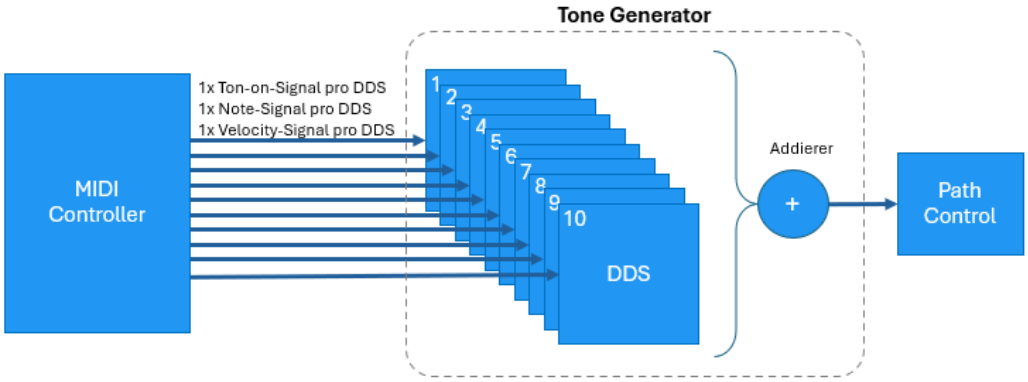


Abbildung 3.2: DDS-Systemarchitektur

Der MIDI-Controller sendet drei Steuersignale an jedes DDS-Modul: ein Ton-on-Signal, das anzeigt, ob eine Note gespielt wird, ein Note-Signal, das die zu spielende Note festlegt, und ein Velocity-Signal, das die Lautstärke beziehungsweise Anschlagstärke der Note steuert. Diese Steuersignale bilden die Grundlage für die Klangerzeugung durch die DDS-Module.

Jede Instanz eines DDS kann einen einzelnen Ton erzeugen. Die Form des Tons wird durch eine vordefinierte Look-Up-Table (LUT) bestimmt, die eine Liste von Zahlenwerten enthält. Diese Werte steuern Frequenz, Periodenform und Amplitude des Tons. Der MIDI-Controller wählt den zu erzeugenden Ton aus, indem er die entsprechende LUT an den DDS übergibt. Der DDS durchläuft dann die Werte in der LUT entsprechend dem vorgegebenen Takt und gibt sie sequenziell aus.

Es gibt insgesamt zehn DDS-Module, die parallel arbeiten und spezifische Frequenzkomponenten bearbeiten. Jedes Modul empfängt die Steuersignale vom MIDI-Controller und erzeugt darauf basierend eine Wellenform.

Die Ausgänge der DDS-Module werden durch einen Addierer kombiniert, der die verschiedenen Frequenzkomponenten der DDS-Module zu einem einzigen komplexen Gesamtsignal integriert. Dieses kombinierte Signal wird anschliessend an die Path Control weitergeleitet, die die weitere Verarbeitung und Ausgabe des Signals übernimmt.

## 3.4 Path Control

Der Path Control Baustein bietet die Möglichkeit, zwischen zwei Signalen, die als Quelle für den WM8731 dienen sollen, umzuschalten. Die Signale sind parallel ausgeführt und werden jeweils von der DDS oder den Parallelen Daten des ADC vom WM8721 bereitgestellt. Somit gibt es die Möglichkeit, einen Digital Loopthroug zu machen oder die DDS als Quelle zu nutzen.

## 3.5 $I^2S$

Der $I^2S$-Bus (Inter-IC Sound) ist ein Standard für die serielle Übertragung von Audiodaten zwischen integrierten Schaltkreisen. Dieser Datenbus sorgt dafür, dass die digital erzeugten Audiosignale in einem Format übertragen werden, das vom Audio-Codec verarbeitet werden kann.

## 3.6 Codec-Controller und $I^2C$

Der $I^2C$-Bus (Inter-Integrated Circuit) kommuniziert mit dem WM8731, um verschiedene Parameteter im Codec Chip zu setzen, wie zum Beispiel das Analoge Loopthrough oder das Stummschalten eines Kanals.
Der Codec-Controller hat prinzipiell die Aufgabe, die seriellen Daten des $I^2S$-Buses paralles umzuwandeln und ebenso parallele Daten seriell umzuwandeln. Dies ist nötig, da die DDS parallel die Daten erstellt und der WM8731 Chip die Daten seriell benötigt.

# 4 Implementierung weiterer Funktionen

Neben den im Kapitel 3 genannten grundlegenden Funktionen, die einen reibungslosen Betrieb des Audio-Synthesizers gewährleisten, wurden weitere Funktionen zum Audio-Synthesizer hinzugefügt. Diese Extrafunktionen erweitern die Funktionalität des Synthesizers. Um einen umfassenden Überblick über die erweiterten Fähigkeiten zu geben, die im Rahmen dieser Arbeit entwickelt wurden, werden diese Extrafunktionen in diesem Kapitel beschrieben.

## 4.1 Bluetooth-Modul Integration und Klaviatur-App

Die digitale Klaviatur-App wurde mit Flutter entwickelt. Flutter ist ein Open-Source-Framework von Google, das die Erstellung von nativen Benutzeroberflächen für iOS und Android aus einer einzigen Codebasis ermöglicht.

Die App bietet eine intuitive Benutzeroberfläche, die es dem Nutzer ermöglicht, Klaviertasten zu drücken und MIDI-Signale zu erzeugen, ohne ein physisches MIDI-Keyboard zu benötigen. Diese Signale werden dabei per Bluetooth an ein MIDI-fähiges FPGA gesendet.

Für die Bluetooth-Kommunikation verwendet die App das 'flutter_blue_classic' Paket mit der Version 0.0.2, das Bluetooth Classic unterstützt. Die Nutzung von Bluetooth Classic basiert auf dem vorhandenen Bluetooth-Modul, welches die Bluetooth-Version 2.1 unterstützt.

### 4.1.1 Einschränkungen bei der Plattformunterstützung

Da das vorhandene Bluetooth-Modul ausschliesslich Bluetooth Classic unterstützt, gibt es Einschränkungen hinsichtlich der Plattformunterstützung:

**Android-Unterstützung:** Android bietet umfassende Unterstützung für Bluetooth Classic. Das 'flutter_blue_classic' Paket ermöglicht es, Bluetooth-Verbindungen auf Android-Geräten herzustellen und Daten zu senden.

**iOS-Unterstützung:** iOS unterstützt Bluetooth Classic nur eingeschränkt. Apple bevorzugt Bluetooth Low Energy (BLE). Aus diesem Grund ist eine Bluetooth-Classic-Kommunikation auf iOS-Geräten mit dem verwendeten Paket nicht möglich.

### 4.1.2 Einrichtung der Bluetooth-Verbindung auf Android

In Abbildung 4.1 ist die Einrichtung einer Bluetooth-Verbindung auf einem Android-Gerät dargestellt. Die Einrichtung einer Bluetooth-Verbindung auf einem Android-Gerät umfasst mehrere Schritte:

1. **Initialisierung:** Initialisierung des Bluetooth-Moduls sowie Überprüfung notwendigen Berechtigungen.

2. **Gerätescan:** Die App scannt nach verfügbaren Bluetooth-Geräten, die im Namen 'Fire' enthalten. Der Nutzer wählt das gewünschte Gerät aus.

3. **Verbindung herstellen:** Nach der Auswahl wird versucht eine Verbindung zum Gerät aufzubauen.

4. **Datenaustausch:** Nach erfolgreicher Verbindung werden die MIDI-Signale über Bluetooth an das verbundene Gerät übertragen und vom FPGA weiterverarbeitet.

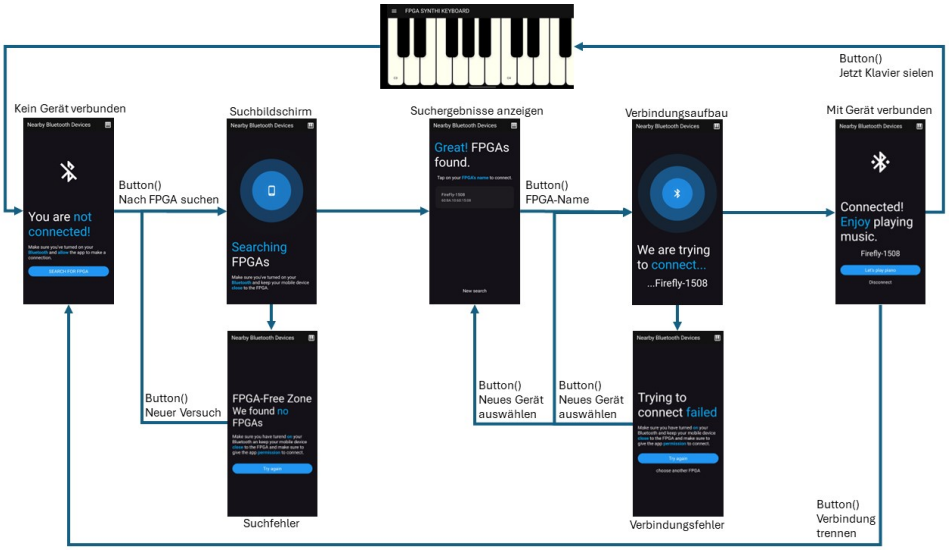


ABBILDUNG 4.1: App-Storyboards mit dem Verbindungsaufbau zum Bluetooth-Modul

## 4.2 FM-Synthese

Die FM-Synthese (Frequenzmodulationssynthese) nutzt zwei unabhängige Direct Digital Synthesis (DDS) Module, die mit VHDL programmiert wurden. Dieses Verfahren ermöglicht es, die Frequenz eines Trägersignals dynamisch mit einem Modulationssignal zu verändern und dadurch komplexe Klangwellenformen zu erzeugen. Diese Methode wird in verschiedenen musikalischen und technischen Anwendungen eingesetzt.

### 4.2.1 Technische Umsetzung

Im Zentrum des FM-Synthese-Systems steht die VHDL-Entität tone_gen, die als primärer Controller für die FM-Synthese fungiert. Die Entität empfängt Eingangssignale über eine 6-MHz-Taktfrequenz und ein Reset-Signal sowie digitale Steuersignale, die die Betriebsmodi der DDS-Module definieren. Diese Steuersignale beinhalten Signale zur Bestimmung, welche DDS-Module aktiv sind, zur Steuerung der Berechnung der nächsten Summe und Arrays, die Informationen zur Tonstärke und zur Note enthalten. Ein Schalter ermöglicht die Änderung des FM-Verhältnisses und der Modulationstiefe.

### 4.2.2 Konfiguration der Frequenzmodulation

Die FM-Parameter wie das Verhältnis (Ratio) und die Tiefe (Depth) der Modulation werden durch einen Prozess gesteuert, der auf die Position des Schalters reagiert. Dieser Prozess passt die FM-Ratio und Depth entsprechend der gewählten Einstellung an, was es ermöglicht, die Klangfarbe und Textur der erzeugten Töne zu variieren. Die Zuweisung der FM-Parameter zu den Modulen erfolgt durch die Erzeugung von Instanzen der Komponente DDS_FM_Dual in einer Schleife, wobei jeder Instanz spezifische Parameter für die Synthese übergeben werden.

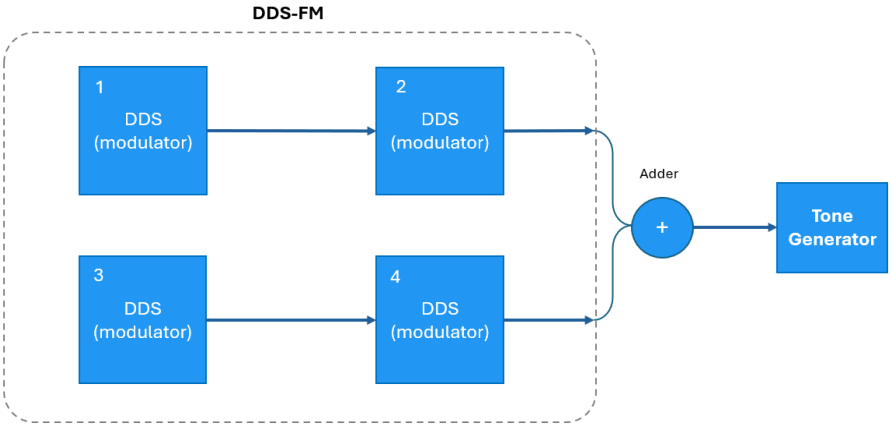


Abbildung 4.2: DDS_FM_DUAL

### 4.2.3 Klangsynthese und Ausgabe

Innerhalb der tone_gen-Architektur werden die von den DDS-Modulen generierten Audiosignale in einem Array gespeichert. Diese Signale werden dann in einem kombinatorischen Prozess aggregiert, um eine komplexe Ausgabewelle zu erzeugen, die dann über die Ausgangsports ausgegeben wird. Die Summation erfolgt in einem Register, das auf Änderungen des Steuerungssignals reagiert und die Summe zur nächsten Taktflanke aktualisiert, um die Kontinuität der Audioausgabe zu gewährleisten.

### 4.2.4 Zusammenfassung der Systemfunktion

Das FM-Synthese-System im FPGA bietet durch die flexible und dynamische Konfiguration der Modulationsparameter und die effiziente Umsetzung in VHDL eine Plattform für die Erzeugung hochwertiger, komplexer Klangwelten. Die modulare Struktur und die präzise Steuerung der individuellen Komponenten ermöglichen es, eine breite Palette von Sounds und musikalischen Effekten zu erzeugen. Durch die Kombination von digitaler Signalverarbeitung mit traditionellen musikalischen Parametern eröffnet das System neue Möglichkeiten für das Design und die Kreation von Audioinhalten.

## 4.3 VU-Meter

### 4.3.1 Überblick konstruktion VU-Meter

Die Umsetzung des VU-Meters erfolgte in zwei Schritten: zunächst die schematische Erfassung der elektrischen Verbindungen, anschliessend die Konstruktion des PCBs auf Basis des Schemas. Dies wurde mit dem Open-Source-Programm KiCad durchgeführt. Die Schaltung wurde im Editor erfasst, in dem alle Verbindungen zwischen den Bauelementen definiert wurden. Danach wurde das Schema im Router-Editor in ein PCB umgesetzt, wobei besonderer Wert auf die Ästhetik gelegt wurde, da das PCB im Endprodukt sichtbar ist. Ergänzend wurde eine Konstruktion in Fusion 360 erstellt und 3D-gedruckt. Nach der mechanischen Fertigung des VU-Meters wurde der Code in VHDL geschrieben. Das Schema und das PCB-Layout des VU-Meters sind im Anhang enthalten.

### 4.3.2 Schema PCB

Um die Daten einfach und ohne zahlreiche GPIOs des FPGA-Boards zu übertragen, wurden auf dem PCB Schieberegister verwendet. Dies ermöglicht, Daten mit nur drei Leitungen ins Schieberegister zu schreiben. Zusätzlich wurden kaskadierbare Schieberegister gewählt, um mehr Ausgänge anzusteuern. Für die Ausgabe wurden pro Kanal 16 rechteckige LED-Elemente genutzt, da diese eng angeordnet werden können und somit die typische Lautstärkeskala bilden. Die Spannung von 5 V auf 3.3 V wurde mittels eines linearen Reglers gesenkt, um die Logiklevel der Signale zu erhalten. Alle Eingänge der Schieberegister wurden herausgeführt, um spätere Flexibilität im Programm zu gewährleisten. Die Ausgänge der Schieberegister wurden über Widerstände an die LEDs angeschlossen, wobei der Widerstandswert niedrig gewählt wurde, um eine hohe Helligkeit zu erzielen, die später mit Pulsweitenmodulation gesenkt werden kann. Der linke und rechte Kanal sind identisch aufgebaut. Zusätzlich wurde ein Encoder ins Schema aufgenommen, um verschiedene Parameter zu verstellen. Die Signale sind über PullUp-Widerstände an die 3,3 V-Leitung angeschlossen, um das Auslesen der Signale zu ermöglichen.

### 4.3.3 Routing PCB

Anschliessend an das Erfassen des Schemas wurde das PCB im Editor geroutet. Alle Komponenten wurden systematisch und in einheitlichen Abständen Angeordnet, ebenso wurde auf eine klare Struktur geachtet, die simpel und elegant sein soll. Dies erleichterte ebenfalls die Übersicht, die später beim Löten von Vorteil ist. Alle Signal-Leiterbahnen wurden mit der Breite 0.4 mm geroutet und der gleiche Abstand wurde überall eingehalten. Leiterbahnen, die zur Stromversorgung dienen, wurden mit 0.8 mm Breite geroutet.
Anschliessend wurde ein Gehäuse mit Fusion 360 für das PCB konstruiert, das anschliessend auf einem 3D-Drucker gedruckt wurde.

### 4.3.4 VHDL Code für VU-Meter

Der VHDL-Code ist in drei Blöcke unterteilt. Der erste Block enthält einen Tiefpassfilter erster Ordnung, der das eingehende Audiosignal verlangsamt und schnelle Schwankungen reduziert, um einen eleganteren Anstieg der Anzeige zu ermöglichen. Anschliessend wird das Signal logarithmisch skaliert, um die wahrgenommene Lautstärke besser widerzuspiegeln, wobei ein If-Statement den gefilterten Signalpegel in Bitmuster umwandelt. Ein Matlab-Programm

unterstützt die Erstellung der Bitwerte für die korrekte Darstellung. Der zweite Block speichert höhere Signalpegel und überschreibt kleinere Werte. Wenn keine grösseren Pegel erkannt werden, erfolgt ein langsames Rücksetzen durch Rechtsverschieben der Werte. Dies sorgt dafür, dass das VU-Meter Spitzenwerte anzeigt, bevor sie langsam verblassen. Der dritte Block schiebt die Daten seriell in das 74HC595-Schieberegister. Ein Modulo-Divider reduziert die Clock-Frequenz von 6.25 MHz auf etwa 100 kHz, um eine zuverlässige Datenübertragung sicherzustellen. Zusätzlich enthält dieser Block einen PWM-Generator zur Steuerung der LED-Helligkeit, wobei die Helligkeit durch den Encoder verändert oder ausgeschaltet werden kann. Die Eingänge wurden in VHDL entprellt, um eine fehlerfreie Bedienung zu gewährleisten.

## 4.4 Siebensegmentanimation

Die Animation gibt durch die Segmente und LEDs den Rückgabewert des Tons wieder. Die Animation besteht aus vier Dateien: vhdl-hex2sev, Siebensegmentanimation, Siebensegment-Top und mux12to6.

### 4.4.1 Aufbau und Funktionsweise

In der Datei Siebensegmentanimation befindet sich der gesamte Code für die Animation, der an die Segmente gesendet wird. Hier werden verschiedene Animationszustände durchlaufen, um die Segmente entsprechend anzusteuern. Die Animation wird durch einen Zähler gesteuert, der die Position innerhalb der Animationssequenz bestimmt. Der Zähler erhöht sich mit jedem Taktzyklus und durchläuft verschiedene Zustände, die festlegen, welches Segmentmuster angezeigt wird. Ähnlich verhält es sich mit der gegebenen Datei vhdl-hex2sev, die ebenfalls die Segmente ansteuert. Dieses Modul wandelt ein 4-Bit-Hexadezimal-Signal in ein 7-Segment-Display-Signal um und sorgt dafür, dass die entsprechenden Ziffern oder Buchstaben auf dem Display dargestellt werden. Um beiden Codes den Zugriff auf die Segmente zu ermöglichen, wurde die Datei Siebensegment-Top hinzugefügt. Diese Datei integriert Siebensegmentanimation und vhdl-hex2sev und steuert die Anzeige auf den 7-Segment-Displays. Zusätzlich enthält Siebensegment-Top einen Multiplexer (mux-12to6), der zwischen den Ausgaben dieser Module umschaltet. Der Multiplexer ermöglicht es, dass die sechs Ausgänge der Siebensegmentanimation sowie die sechs Ausgänge der vhdl-hex2sev auf sechs gemeinsame Ausgänge zusammengeführt werden. Die jeweiligen zwölf Ausgänge werden auf den Multiplexer geführt, der diese dann als Eingänge verwendet. Im Multiplexer wird zwischen den beiden Gruppen von sechs Eingängen mit einem Schalter umgeschaltet, sodass die gewünschten sechs Ausgänge direkt auf die Segmente gehen.

### 4.4.2 Ergebnis

Die Animation zeigt eine vorwärts und rückwärts laufende Sequenz von Mustern auf den Segmenten. Diese Sequenz besteht aus verschiedenen Zuständen, in denen nacheinander die Segmente aktiviert werden, um eine laufende Lichtfolge darzustellen. Bei der Vorwärtsbewegung beginnt die Animation mit einem Segment und läuft durch den Zähler von 1 bis 16. Nach Erreichen des Endes der Sequenz kehrt die Animation um und läuft rückwärts durch den Zähler, was den Effekt einer kontinuierlich fliessenden Bewegung der LEDs erzeugt.

# 5 Schlussfolgerung

## 5.1 Verbesserungen und Erweiterungen

Zusätzlich besteht die Möglichkeit, Daten vom FPGA an die App zu senden. Beispielsweise kann das FPGA mit einem Keyboard verbunden werden, sodass die jeweils gedrückten Tasten in Echtzeit an die App übermittelt werden. Dies könnte für Lern- und Übungszwecke, wie das Erlernen von Noten oder das Einüben von Musikstücken, dienen. Die App kann die empfangenen Daten analysieren und dem Benutzer Rückmeldungen geben.

Eine zusätzliche Erweiterung der MIDI-Funktionalität wäre das Hinzufügen verschiedener Instrumente. Jedes Instrument würde seinen eigenen MIDI-Kanal besitzen, zwischen denen man umschalten kann.

## 5.2 Fazit

Das vorliegende Projekt demonstrierte, dass mit einem MIDI-Controller viel mehr als nur ein einzelner Ton gesteuert werden kann. Die Bedienung von UART-Schnittstelle, I2C sowie I2S konnte erlernt werden. Auch die erweiterten Projekte lehrten das Vorausdenken und das Setzen von Zielen für verschiedene Funktionen.

Durch die Implementierung verschiedener Module wie der FM-Synthese und des VU-Meters wurden praktische Erfahrungen in der digitalen Signalverarbeitung gesammelt. Die Integration von Bluetooth-Kommunikation und die Entwicklung einer intuitiven Benutzeroberfläche in der App erweiterten die Fähigkeiten in der App-Entwicklung und der drahtlosen Datenübertragung.

Insgesamt vermittelte dieses Projekt nicht nur technische Kenntnisse, sondern auch wichtige Projektmanagementfähigkeiten. Die erfolgreiche Umsetzung der geplanten Funktionen und die Lösung auftretender Probleme zeigten, wie wichtig eine gründliche Planung und flexible Anpassung sind.

Alle vorgegebenen Meilensteine wurden erreicht und zu den festgesetzten Terminen abgegeben und präsentiert. Die erweiterten Funktionen wurden ebenfalls erreicht und konnten dem Dozenten in der letzten Woche vorgeführt werden. Somit wurde der Synthesizer mit allen vorgegebenen Zielen abgeschlossen.

# A Anhang

## A.1 Schemata und PCB-Layout

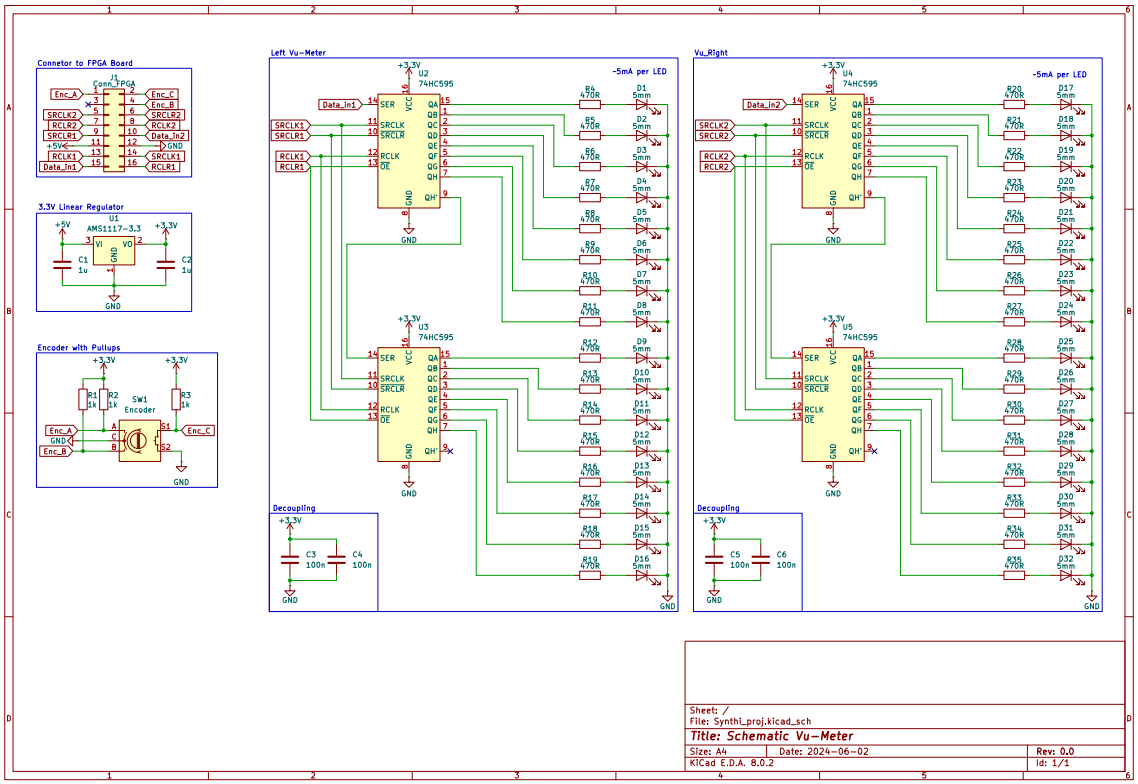


Abbildung A.1: Schema des Vu-Meters

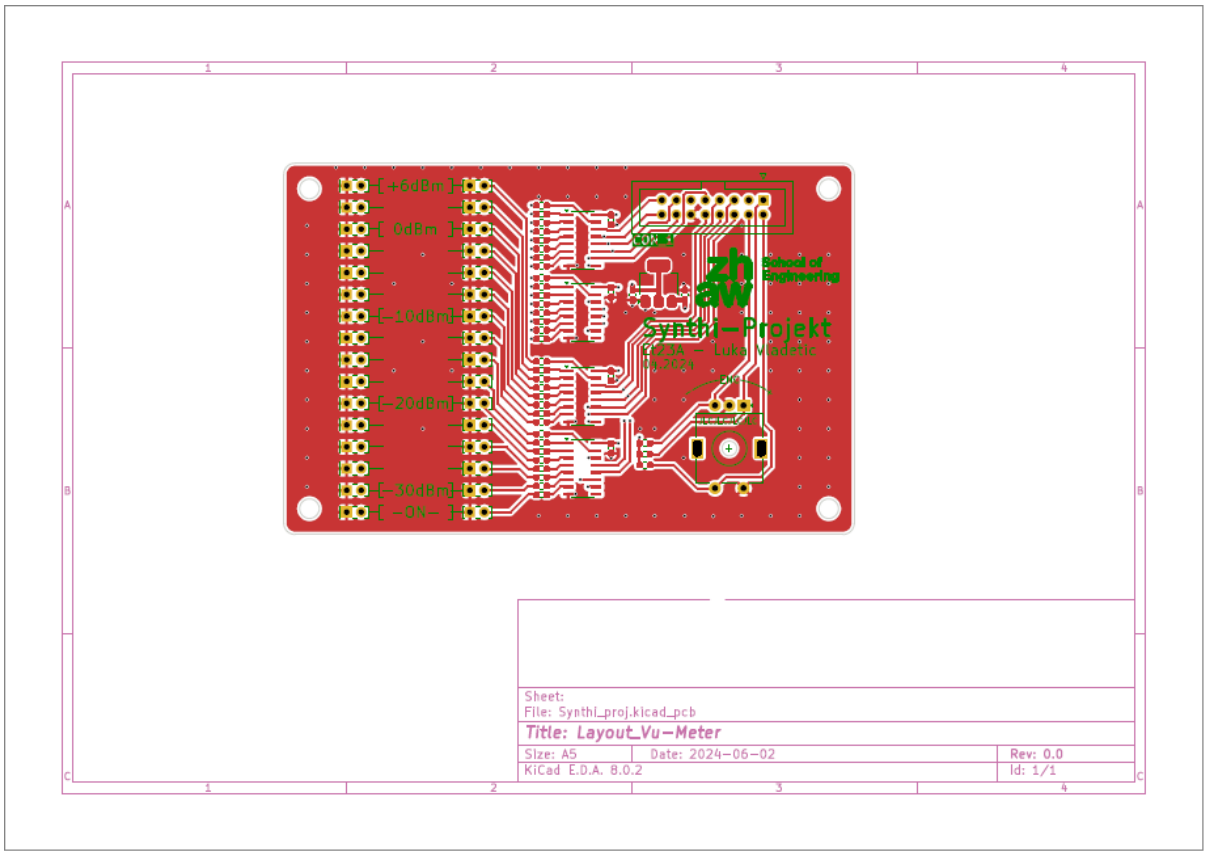


Abbildung A.2: Obere Lage und Aufdruck

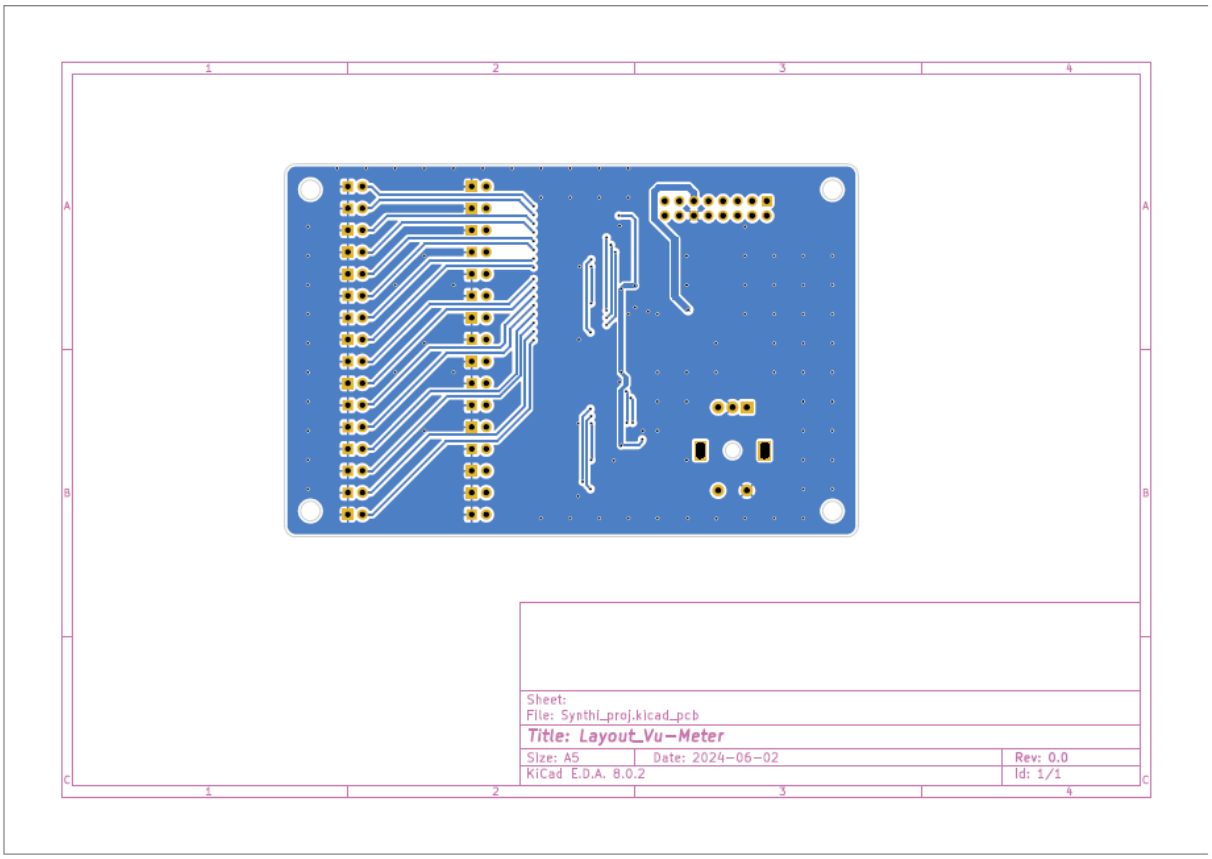


Abbildung A.3: Untere Lage

# B Anhang

## B.1 Übersicht der Schalterstellungen für verschiedene Betriebsmodi

Dieser Anhang bietet eine Übersicht über die verschiedenen Schalterstellungen zur Konfiguration und dem Betrieb der Modi des FPGAs. Die Tabellen sind in vier Hauptkategorien unterteilt: Codec-Initialisierung, MIDI-Input, vordefinierte FM-Synthese-Einstellungen und weitere Funktionen. Jede Tabelle zeigt die möglichen Schalterstellungen und die entsprechenden Funktionen. Diese Informationen sind für die korrekte Einrichtung und Nutzung des Systems wichtig.

Tabelle B.1: Schalterstellungen für die Codec-Initialisierung

| SW(2) | SW(1) | SW(0) | Funktion |
|---|---|---|---|
| 0 | 0 | 1 | Analog-Bypass |
| 1 | 0 | 1 | Analog-Mute Links |
| 0 | 1 | 1 | Analog-Mute Rechts |
| 1 | 1 | 1 | Analog-Mute Beide |
| x | x | 0 | ADC/DAC 0dB 48K |

Tabelle B.2: Schalterstellungen für MIDI-Input

| SW(5) | SW(4) | Input |
|---|---|---|
| 0 | 0 | GPIO26 (MIDI-Keyboard) |
| 0 | 1 | Bluetooth |
| 1 | 1 | USB |

Tabelle B.3: Schalterstellungen für vordefinierte FM-Synthese-Einstellungen

| SW(8) | SW(7) | SW(6) | Ratio1 | Depth1 | Ratio2 | Depth2 |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 011 | 100 | 010 | 001 |
| 0 | 0 | 1 | 100 | 110 | 001 | 011 |
| 0 | 1 | 0 | 101 | 101 | 110 | 010 |
| 0 | 1 | 1 | 110 | 011 | 101 | 100 |
| 1 | 0 | 0 | 111 | 111 | 000 | 000 |

Tabelle B.4: Schalterstellungen für weitere Funktionen

| SW(9) | SW(3) | KEY(0) | Funktion |
|---|---|---|---|
| 0 | x | 1 | Siebensegment-Anzeige |
| 1 | x | 1 | Anzeige der gespielten Note und Velocity |
| x | 1 | x | Loopback-Modus |
| x | x | 0 | System-Reset |