KU LEUVEN

FACULTEIT INGENIEURSWETENSCHAPPEN

Systeemontwerp met HDL

Audio spectrum analyzer met HDMI uitvoer

Gert-Jan Andries
Xavier Dejager
Nick Steen

Master of Science in de industriële ingenieurswetenschappen: elektronica-ICT, optie Elektronica

Docent:
Sammy Verslype

Academiejaar 2014 – 2015

Inhoudsopgave

Afkortingen en symbolen **iii**

1 Inleiding **1**

2 Spectrum Analyzer **2**
2.1 Audio . . . 2
2.2 Analyse . . . 2
2.3 Hamming Window . . . 2
2.4 Uitgang . . . 3

3 Overzicht **4**
3.1 Input . . . 4
3.2 Output . . . 4
3.3 Mapping . . . 5
3.4 Process . . . 6
3.5 Gebruikte IPs . . . 6
3.6 ZedBoard Pinout . . . 7

4 Componenten **9**
4.1 Audio interface . . . 9
4.2 Memorycontrol . . . 11
4.3 Delay . . . 11
4.4 OLED Top . . . 12
4.5 VGA HDMI . . . 16

5 Besluit **21**
5.1 Sampelen van audio . . . 21
5.2 Dataverwerking en FFT . . . 21
5.3 Spectrum visualiseren . . . 21
5.4 OLED display . . . 22
5.5 VHDL toolbox . . . 22
5.6 Algemeen . . . 22

A Overzicht spectrum analyzer **24**

B Componenten **25**

B.1 Top 25
B.2 Audio interface 26
B.3 Delay 27
B.4 Oled top 28
B.5 Oled initialize 29
B.6 Oled text 30
B.7 SPI control 31

C ADAU1761 Settings **32**

Afkortingen en symbolen

Afkortingen

ADC	Analoog - Digitaalconverter
ASCII	American Standard Code for Information Interchange
AXI	Advanced Extensible Interface
CS	Chip Select
DFT	Discrete Fourier Transformatie
DSP	Digital Signal Processing
FFT	Fast Fourier Transformatie
FPGA	Field-Programmable Gate Array
HDL	Hardware Description Language
HDMI	High Definition Multimedia Interface
I^2C	Inter Integrated Circuit bus
I^2S	Inter Integrated Circuit Sound
IP	Intellectual Property
LED	Light Emitting Diode
LSB	Least Significant Bit
MSB	Most Significant Bit
OLED	Organic Light Emitting Diode
PLL	Phase-Locked Loop
RAM	Random Acces Memory
RGB	Red Green Blue / Rood Groen Blauw
ROM	Read Only Memory
SCL	Serial Clock
SDA	Serial Data
SPI	Serial Peripheral Interface
VGA	Video Graphics Array
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
YCbCr	Luminance Chrominance

Hoofdstuk 1

Inleiding

Tijdens de lessenreeks systeemontwerp met HDL werd kennis gemaakt met de Xilinx Vivado ontwikkelomgeving. Onder meer volgende topics kwamen aan bod:

Simulatie en constraints: In dit hoofdstuk werden constraints en simulaties in de Vivado omgeving behandeld. Hiernaast kwam ook het ZedBoard ontwikkelbord aan bod.

IP en clocking resources: In dit hoofdstuk werd dieper ingegaan op IP's en hoe deze kunnen worden geïntegreerd. Verder werd ook gekeken naar klokpaden binnen een ontwerp en het gebruik van de clocking wizard.

Hardware debugging: In deze les werd uitgelegd hoe men door middel van een ILA uit de IP catalog een signaal kan monitoren dat zich binnen de FPGA bevindt.

Na deze theorielessen werd in groep een groter project uitgwerkt op een FPGA-ontwikkelbord (ZedBoard). Dit project bestond eruit om een audio spectrum analyzer te maken. Deze analyzer neemt als invoer een stereo audiosignaal dat wordt binnengelezen via een audiocodec. Vervolgens dient er een spectra van deze signalen berekend te worden op de FPGA (zowel voor de linker als het rechter audiokanaal). Dit spectrum zal dan op een beeldscherm worden weergegeven door middel van een HDMI interface.

Het project kan worden aangevuld met volgende extra features:

- Piek aanduiding (bv. ander kleur of horizontale lijn)
- Werken met verschillende kleuren (i.f.v. frequentie of amplitude)
- Keuze aan gebruiker over grafiektype (bars, dots, lijn...)
- ...

In dit verslag zal de werking en opbouw van het project overlopen worden. Elke component zal kort worden besproken en ook de moeilijkheden tijdens het ontwerpproces komen aan bod.

Hoofdstuk 2

Spectrum Analyzer

2.1 Audio

Audio is een continu variabel en analoog signaal, dat dus niet direct digitaal te verwerken valt. Het is vaak niet-periodiek en kan dus niet per periode geanalyseerd worden. Om het digitaal te kunnen verwerken, wat uiteindelijk gebeurt bij de interne werking van de FPGA, dient het gesampled te worden, om daarna via een ADC aan de invoer van de digitale schakeling gelegd te worden.

2.2 Analyse

De analyse gebeurt digitaal en via de Fourier transformatie. De Fourier transformatie baseert zich op het feit dat een signaal kan opgedeeld worden in de som van sinussen en cosinussen. Elke sinus of cosinus heeft een eigen frequentie en amplitude, waardoor er dus kan geanalyseerd worden welke frequenties er sterker aangwezig zijn in het ingangssignaal. Gezien het karakter van de audio dient hier met een DFT gewerkt te worden, een transformatie die een eindige periode nodig heeft om een analyse op uit te voeren. Door over een vaste periode te samplen, wordt een window gegenereerd die zo als input kan dienen voor deze transformatie.

Gezien de digitale aard van de schakeling en de snelheid die nodig is om de verwerking uit te voeren, is het beter een aantal opeenvolgende samples te kiezen die een macht van 2 zijn. Dit houdt in dat er 2, 4, 8 ... samples genomen dienen te worden. Zo kan dan via een sneller algoritme, de FFT, het resultaat berekend worden.

2.3 Hamming Window

Omdat we een FFT willen toepassen in real time moeten we het signaal in stukken hakken. Hoe groter dit stuk hoe beter de lage frequenties kunnen bepaald worden, maar hoe meer delay er komt op de uitgang van het spectrum. En door de aard van het FFT algoritme zal dit stuk als een periodiek herhalend stuk aanzien worden. Nu weten we dat een plotste verandering in amplitude veel hoge spectrale componenten zal bevatten. En omdat het sampelen op een bepaald tijdstip gebreurd kan dit

er voor kan zorgen dat het eerste en het laatste sample niet gelijk uitkomen. En mogelijks zelf veel van elkaar verschillen. Daar zal dus een scherpe overgang zijn. Dit effect zal het spectrum aanzienlijk beïnvloeden. Om dit tegen te gaan maken we gebruik van een hamming window. Dit zorgt er voor dat de uiteinden heel geleidelijk naar nul gebracht worden, daardoor is het ommogelijk om daar die plotse overhang te krijgen. Het hamming window zelf heeft ruwweg de vorm van een cosinus:

$$w(n) = \alpha - \beta \cos\left(\frac{2\pi n}{N-1}\right) \tag{2.1}$$

Door de samples te convolueren met de het punten van het hamming window bereiken we dat de uiteinden nagenoeg nul zijn op de uiteindes. Merk wel op dat wij de waardes voor de hamming functie gemapt hebben van 0 tot en met 255. Dit omdat het veel eenvoudiger is om integers te vermenigvuldigen dan floating point getallen te vermenigvuldigen op een FPGA.

2.4 Uitgang

Uit de uitgang van de FFT kan dan gehaald worden hoe sterk bepaalde frequenties aanwezig zijn. Om dit aan de gebruiker duidelijk te maken, kan dit via een grafische output weergegeven worden.

Hoofdstuk 3

Overzicht

De toplaag combineert alle onderliggende lagen om een functioneel geheel te verkrijgen. Als inputs heeft de toplaag enkel objecten waarop de gebruiker effectief kan interageren zoals knoppen of een audio aansluiting. Ook de outputs zijn direct merkbaar door de gebruiker, zijnde een HDMI of OLED scherm, of LEDs om de status aan te duiden.

3.1 Input

Door middel van 6 schakelaars en een audio interface is de gebruiker in staat om de inputs zelf aan te passen. Door gebruik van de schakelaars kan de voorstellingswijze van het spectrum op het HDMI scherm gekozen worden. Via een resetknop kunnen de verschillende componenten terug ingesteld worden op hun initiële waarde.

Doordat er gewerkt wordt met een externe geluidsbron is het mogelijk om eender welke audio af te spelen via de 3.5mm audio jack. Hierin voert men een signaal toe dat afkomstig kan zijn van een PC of mp3-speler,

3.2 Output

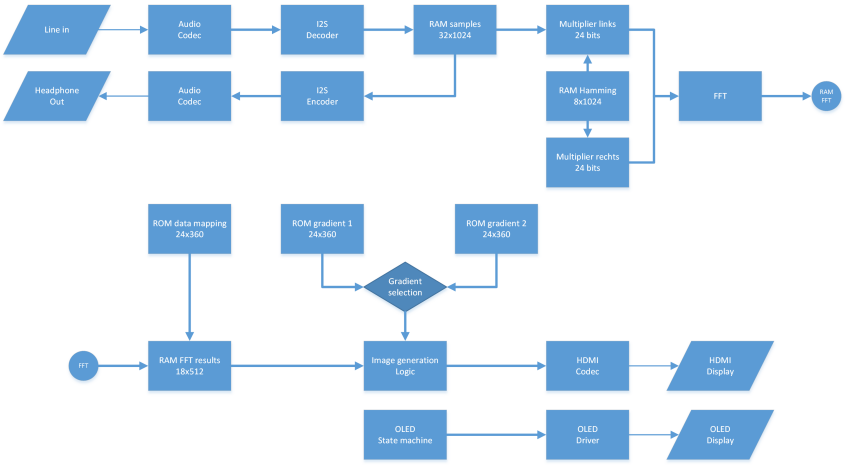
Op het OLED scherm dat aanwezig is op het ZedBoard wordt weergegeven wat de huidige instellingen zijn. Deze instellingen worden aangepast door middel van de schakelaars. Tijdens de opstartfase wordt een vaste text weergegeven gedurende 4 seconden. Hierna wordt overgegaan naar de modus met die de instellingen weergeeft.

Via de HDMI uigang wordt het frequentiespectrum weergegeven op een HDMI-compatibel scherm.

De binnengelezen audio zal ook weer naar buitengebracht worden zodat de gebruiker via de koptelefoon in staat is de audio te beluisteren die bij het weergegeven frequentiespectrum hoort.

Wanneer een schakelaar van postie veranderd zal ook het bijhorend ledlampje oplichten. Hierbij is ook rekening gehouden met de onderlinge afhankelijkheid tussen verschillende modi.

3.3 Mapping



Figuur 3.1: Globaal overzicht van de audio equalizer

In figuur 3.1 is een globaal overzicht weergegeven van de verschillende componenten en hoe deze verbonden zijn met elkaar. Een grotere versie van deze figuur is toegevoegd in appendix A op bladzijde 24.

1. Een audiosignaal wordt aangelegd aan de lijningang van het ZedBoard.
2. Het audiosignaal wordt binnengelezen door middel van de audiocodec ADAU1761.
3. Het audiosignaal wordt gedecodeerd door middel van een I^2S decoder
4. Een RAM geheugen wordt gevuld tot er 1024 audiosamples in aanwezig zijn. Elk sample bestaat uit 16-bit audiodata per kanaal.
5. Wanneer het audiosample geheugen volledig gevuld is worden de samples een voor een uitgelezen en vermenigvuldgt met een hamming window waarde. Dit gebeurt afzonderlijk voor elk kanaal.
6. De audiosamples worden vervolgens aangevoerd aan de FFT component. Deze leest ook 1024 samples in alvorens de transformatie begint.
7. Wanneer de transformatie klaar is worden de samples opgeslagen in een RAM geheugen. De 512 samples worden geschaald naar een breedte van 1280 pixels.
8. Tot slot zal de Image generation logic afhankelijk van de schakelaars een beeld genereren dat op het scherm wordt weergegeven.

Wanneer de gebruiker een schakelaar van positie wijzigt zal op het OLED scherm een passende tekst worden weergegeven die aantoont welke modus er actief is.

3.4 Process

Bij de opstart van het ZedBoard zal een state-machine doorlopen worden. In een eerste fase zal het OLED scherm worden geïnitialiseerd. In een tweede fase zal gedurende 4 seconden een vaste tekst op het OLED scherm weergegeven worden. Vervolgens wordt overgegaan naar een derde fase waar een andere tekst ook 4 seconden wordt weergegeven. Wanneer deze 8 seconden voorbij zijn wordt er gereageerd op de gebruikersinvoer. Het OLED scherm zal vanaf dan de actuale status weergeven van de schakelaars. Op het HDMI scherm wordt continu het spectrum van de binnenkomende audio weergegeven.

Het process in het top level voorziet ook in de vertraging van verschillende signalen die worden doorgegeven aan de verschillende systeemcomponenten zodat de data steeds op het juiste moment bij de component toekomt. Een overzicht van de hiervoor gevolgde statemachine is terug te vinden in bijlage B.1.

3.5 Gebruikte IPs

clk_wiz_0: (Clocking Wizard) Deze IP staat in voor het genereren van de klokken die nodig zijn binnen het project. Er wordt een klok van 100 MHz en 12.288 MHz gegenereerd.

audio_sample_block_mem: (Block Memory Generator) In dit RAM geheugen worden de audiosamples opgeslagen die binnengelezen worden door middel van de audiocodec. Door middel van een adresgenerator worden de in- en uitleesadressen voor dit geheugen gegenereerd. De invoer van data is rechtstreeks afkomstig van de audiocodec. De uitvoer gaat via de hamming window multiplier naar de FFT.

block_mem_hamming: (Block Memory Generator) Dit ROM geheugen zijn de waarden voor het hamming window opgeslagen. Het adres dat gebruikt wordt om het audio sample geheugen uit te lezen zal 2 kloktikken vertraagd worden zodat het ook kan gebruikt worden om het adres van dit geheugen te genereren. De uitvoer van dit geheugen (de hamming window waarde voor het overeenkomstig samplenummer) zal worden aangelegd aan de hamming window multiplier.

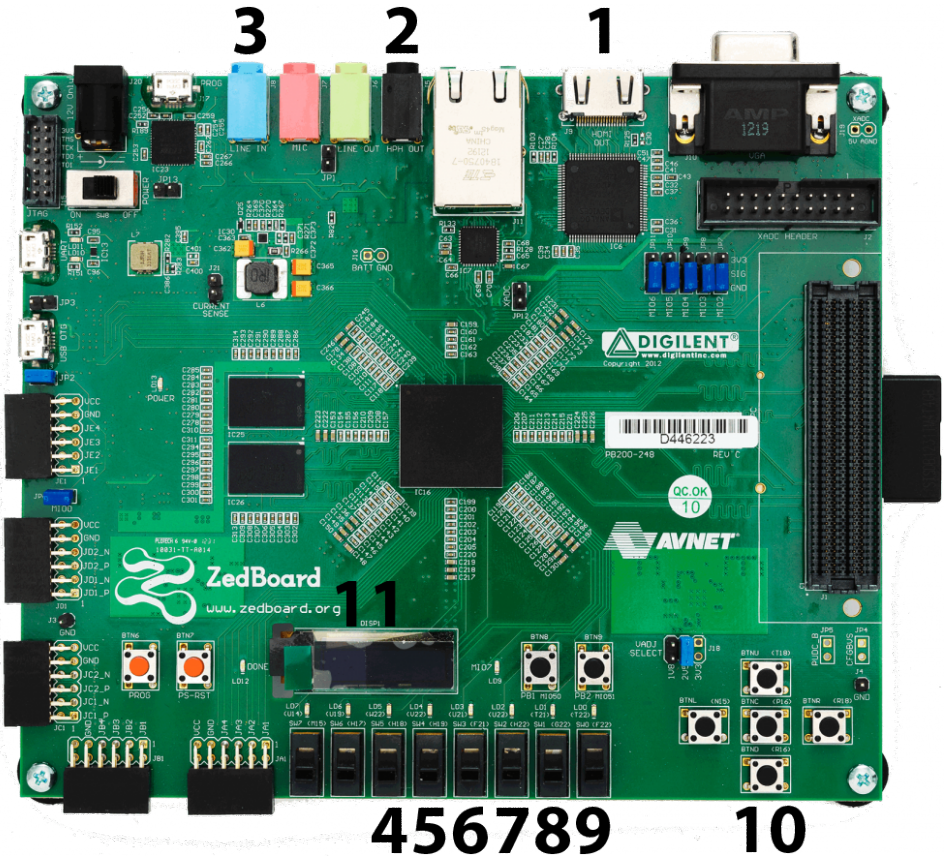
hamming_multiplier_left: (Multiplier) Deze component zorgt voor het vermenigvuldigen van het linker kanaals audiosample met de waarde van het hamming window. Vervolgens wordt de data aangevoerd aan de FFT.

hamming_multiplier_right: (Multiplier) Deze component zorgt voor het vermenigvuldigen van het rechter kanaals audiosample met de waarde van het hamming window. Vervolgens wordt de data aangevoerd aan de FFT.

FFT_LR: (FFT) Deze component voert de FFT transformatie uit op 1024 samples. Deze samples zijn afkomstig uit het audio sample geheugen, maar zijn eerst vermenigvuldigd met een hamming window. Wanneer aangegeven wordt dat het laatste sample is binnen gelezen start de FFT component met het berekenen van het spectrum. Wanneer deze klaar is wordt een FIN-vlag op hoog gezet. Vervolgens wordt deze data opgeslagen in het FFT output geheugen.

FFT_output_mem: (Block Memory Generator) In dit geheugen wordt de uitvoer van de FFT opgeslagen. Wanneer de HDMI VGA nieuwe data nodig heeft leest deze dit geheugen uit. Vervolgens kan op het scherm het spectrum weergegeven worden.

3.6 ZedBoard Pinout



Figuur 3.2: Weergave van de gebruikte componenten op het ZedBoard

HDMI uitvoer: Via deze aansluiting kan een HDMI scherm aangesloten worden waarop het audiospectrum zal worden weergegeven. [1]

Audiouitvoer naar hoofdtelefoon: De audio kan beluisterd worden door middel veen een hoofdtelefoon die aangesloten wordt op de hph out connectie. [2]

Audioinvoer: Audio wordt aangevoerd via de line-in plug op het ZedBoard. [3]

Schuifknop voor rounded rectangle mode: Hiermee kan men de blokjes uit het discreet spectrum een lichte afronding geven, een aangenaam visueel effect. [4]

Schuifknop voor averaging aan/uit: Met deze schakelaar kan de uitmiddeling in of uitgeschakeld worden. Dit zorgt voor een vloeiender verloop van het spectrum. [5]

Schuifknop voor rectangles aan/uit: Hiermee kan er gewisseld worden tussen een nagenoeg continu spectrum en amplitude en een discreet spectrum en amplitude.[6]

Schuifknop voor thema selectie jet/orange: Hiermee kan gekozen worden tussen 2 gradiënten. De zogenaamde jet kleuren en een oranje thema.[7]

Schuifknop voor edgemode: Met deze schakelaar kan gekozen worden om enkel de buiteste rand van het spectrum volledig te kleuren. De rest wordt dan half zo helder afgebeeld.[8]

Schuifknop voor peakmode: Hiermee kan gekozen worden om het spectrum af te beelden als een punt in plaats van als balk. Dan wordt enkel het bovenste deel van de balk afgebeeld, de rest is de achtergrondkleur.[9]

Resetknop: Deze knop zorgt voor een resetactie in het toplevel. Indien een bepaalde component ook een reset ingang heeft dan wordt deze knop ook met deze ingang verbonden.[10]

OLED scherm: Data over de actieve modus wordt weergegeven op het OLED scherm. Wanneer een modus wijzigt zal ook de tekst worden aangepast. [11]

Hoofdstuk 4

Componenten

4.1 Audio interface

De audio interface-component (audio_if) voorziet de mogelijkheid om de ADAU1761 te configureren door gebruik te maken van de I^2C-databus. Hiernaast is het ook mogelijk om audiosamples te ontvangen en te versturen van en naar de codec door middel van de I^2S-databus. Een overzicht van de hiervoor gevolgde statemachine is terug te vinden in bijlage B.2.

Door middel van configuratieregisters die via de I^2C-databus ingesteld kunnen worden, kan een configuratie van de audiocodec gebeuren. Aan de hand van het programma SigmaStudio van Analog Devices kunnen de correcte registerinstellingen makkelijk bepaald worden. In bijlage C is een overzicht gegeven van deze instellingen.

Signaal	**Type**	**Beschrijving**
clk_100M_in	Clock	100 MHz klokinvoer.
clk_12M288	Clock	12.288 MHz klokinvoer.
m_clk	Clock	gegenereerde masterklok voor ADAU1761.
lr_clk	Clock	gegenereerde left/right klok voor ADAU1761.
b_clk	Clock	gegenereerde bitklok voor ADAU1761.
sdata	Input	seriële data invoer afkomstig van de ADAU1761
audio_sample_in_l	Input	Aanvoer van een 16-bit breed audiosample voor het linker audiokanaal.
audio_sample_in_r	Input	Aanvoer van een 16-bit breed audiosample voor het rechter audiokanaal.
I^2C_addr	Output	Uitvoer van het gegenereerde I^2C-adres voor het instellen van de configuratieregisters.
sample_clk	Output	Uitvoer van de sampleklok. Wordt hoog telkens een nieuw sample beschikbaar is.
sample_l	Output	Uitvoer van een 16-bit breed audiosample voor het linker audiokanaal.
sample_r	Output	Uitvoer van een 16-bit breed audiosample voor het rechter audiokanaal.
serial_data_out	Output	Uitvoer van de seriële data naar de ADAU1761.
sda	Output	Seriële data lijn voor de I^2C-interface.
scl	Output	Seriële klok lijn voor de I^2C-interface.

4.1.1 ADAU1761-interface

De ADAU1761-interface voorziet de mogelijkheid om audiosamples te verzenden en ontvangen. De seriële data die afkomstig is van de audiocodec zal worden omgezet in een 16-bit breed signaal. Daarnaast zal ook het 16-bit brede aangevoerd audiosample omgezet worden naar een serieel signaal.

Signaal	**Type**	**Beschrijving**
lr_clk	Clock	Left/right klokinvoer.
b_clk	Clock	Bitklok invoer.
sdata	Input	Seriële data invoer afkomstig van de ADAU1761.
audio_sample_in_l	Input	Aanvoer van een 16-bit breed audiosample voor het linker audiokanaal.
audio_sample_in_r	Input	Aanvoer van een 16-bit breed audiosample voor het rechter audiokanaal.
sample_l	Output	Uitvoer van een 16-bit breed audiosample voor het linker audiokanaal.
sample_r	Output	Uitvoer van een 16-bit breed audiosample voor het rechter audiokanaal.
sample_clk	Output	Uitvoer van de sampleklok. Wordt hoog telkens een nieuw sample beschikbaar is.
serial_data_out	Output	Uitvoer van de seriële data naar de ADAU1761.

4.2 Memorycontrol

De memorycontrol component staat in voor het genereren van adressen. Deze adressen worden gebruikt om data op te slaan en op te vragen in een blockmemory. De schrijfadressen worden gegenereerd wanneer een nieuw sample binnengelezen wordt. Het proces dat deze adressen genereert wordt getriggerd op de sample_clk_in.

Wanneer er 1024 samples binnengelezen zijn in het geheugen worden deze adressen aangemaakt aan een kloksnelheid van 100MHz om de samples weer uit te lezen en door te geven naar het hamming window en vervolgens de FFT.

Signaal	**Type**	**Beschrijving**
clk_100M	Clock	100 MHz klokinvoer, op deze klok worden de leesadressen gegenereerd.
sample_clk_in	Clock	Klok die aangeeft dat een sample klaar is, op deze klok wordt een schrijfadres gegenereerd.
ready_enable_out	Output	Geeft aan dat de uitvoer klaar is.
last_flag	Output	Geeft aan dat het laatste adres van een reeks adressen (0-1023) gegenereerd is.
write_address	Output	uitvoer van het schrijfadres (10-bit).
read_address	Output	Uitvoer van het leesadres (10-bit).

4.3 Delay

De delaycomponent voorziet de mogelijkheid om na een ingesteld aantal milliseconden (max 4096) een fin-vlag hoog te maken. D.m.v. een state-machine, terug te vinden in bijlage B.3, is het mogelijk om in een state te blijven tot de fin-vlag hoog wordt.

Aan de delaycomponent wordt een klok van 100 MHz aangelegd. Telkens wanneer een milliseconde verstreken is zal een teller verhoogd worden todat het gewenste aantal bereikt is.De verstreken tijd wordt dan aangeduid door een hoge fin-vlag.

Signaal	**Type**	**Beschrijving**
CLK	Clock	De 100 MHz klokinvoer.
RST	Input	Via dit signaal is het mogelijk de delaycomponent te resetten.
DELAY_MS	Input	Dit signaal bevat de instelling van het aantal milliseconden (12-bit breed)
DELAY_EN	Input	Aan de hand van dit signaal wordt de delaycomponent gestart.
DELAY_FIN	Output	Vlag die de verstreken delay-tijd aangeeft.

4.4 OLED Top

De OledTop component staat in voor het initializeren van de OLED door middel van de OledInit component. Daarnaast staat hij ook in voor het weergeven van tekst op het scherm door middel van de OledText component.

Deze component bevat een statemachine met 3 states: Idle, OledInitialize en OledTextState. Waneer de component gestart wordt, zal van Idle overgegaan worden naar OledInitialize. Deze state blijft zolang aangehouden todat de initialisatie volledig voltooid is en neemt ongeveer 100ms in beslag. Hierna wordt overgegaan naar de OledTextState die het mogelijk maakt om tekst naar het OLED scherm te sturen.

Signaal	**Type**	**Beschrijving**
CLK	Clock	Klokinvoer van 100 MHz.
RST	Input	Synchrone reset waarmee de state terug naar Idle gebracht wordt. De initialisatie zal hierdoor opnieuw worden uitgevoerd.
DATA_IN	Input	20-bit brede datainvoer die het nummer bevat van de tekstregel die moet weergegeven worden (5-bit per regel).
ENABLE	Input	Via dit signaal wordt deze component geactiveerd.
SDIN	Output	Seriële data uitvoer, input van OLED.
SCLK	Output	Seriële klok uitvoer, input van OLED.
DC	Output	Data/Command Control.
RES	Output	Power Reset for Controller and Driver.
VBAT	Output	Power Supply for DC/DC Converter Circuit.
VDD	Output	Power Supply for Logic.
INIT_READY	Output	Deze vlag geeft aan dat de initialisatie voltooid is.
TEXT_READY	Output	Deze vlag geeft aan dat de tekst op het scherm getekend werd (korte puls).

4.4.1 OLED Initialize

Om het OLED scherm te kunnen gebruiken moet het eerst geïnitialiseerd worden. De initialisatieprocedure wordt beschreven in de datasheet van de fabrikant. Er dienen verschillende stappen doorlopen te worden waarbij de instellingen naar het OLED scherm verzonden worden. Omdat de OLED een spanning nodig heeft van 7.2V worden chargepumps gebruikt om de 3.3V spanning om te zetten naar 7.2V. Hiervoor dienen op het juiste moment enkele pinnen hoog gemaakt te worden gedurende de initialisatie.

De initialisatieprocedure wordt uitgevoerd door middel van een state-machine. Telkens wanneer een bepaalde actie volbracht is, wordt overgegaan naar de volgende state. Wanneer de initialisatie volledig voltooid is, wordt dit aangegeven door middel van een INIT_READY_OUT-vlag. Een flow chart van deze statemachine is te vinden in appendix B.5.

Signaal	**Type**	**Beschrijving**
CLK	Clock	Klokinvoer van 100 MHz.
RST	Input	Synchrone reset waarmee de state terug naar Idle gebracht wordt. De initialisatie zal hierdoor opnieuw worden uitgevoerd.
EN	Input	Via dit signaal wordt deze component geactiveerd.
CS	Output	Chipselect, niet gebruikt op het ZedBoard.
SDO	Output	Seriële data uitvoer, input van OLED.
SCLK	Output	Seriële klok uitvoer, input van OLED.
DC	Output	Data/Command Control.
RES	Output	Power Reset for Controller and Driver.
VBAT	Output	Power Supply for DC/DC Converter Circuit.
VDD	Output	Power Supply for Logic.
INIT_READY_OUT	Output	Deze vlag geeft aan dat de initialisatie voltooid is.

4.4.2 OLED Text

De OledText component maakt het mogelijk om tekst naar het scherm te schrijven. Deze tekstdata bevindt zich in een geheugen waarin maximaal 32 lijnen tekst van elk 16 karakters opgeslagen kunnen worden. Door middel van een 20-bit brede invoer kunnen er 4 regels geselecteerd worden die naar het scherm geschreven worden. De 5 LSB's geven de eerste tekstregel en de 5 MSB' geven de laatste tekstregel.

Telkens wanneer er een verandering is van DATA_IN, zal de tekst opnieuw naar het scherm geschreven worden. Om de tekst naar het scherm te schrijven, wordt gebruik gemaakt van een state-machine die stap voor stap de nodige acties onderneemt. Een flow chart van deze statemachine is te vinden in appendix B.6. Kortweg worden volgende stappen doorlopen:

1. Wanneer er een verandering is van invoerdata wordt het proces gestart.
2. De tekstdata wordt lijn per lijn opgevraagd uit het geheugen.
3. Vervolgens wordt het scherm overschreven met allemaal blancolijnen.

4. De juiste lijn om naar te schrijven wordt ingesteld (0, 1, 2, 3).
5. De geselecteerde lijn wordt karakter per karakter opgevuld.
    a) Het huidige karakter zal worden opgevraagd uit het karaktergeheugen. Dit geheugen bevat informatie over hoe een karakter naar het scherm geschreven moet worden.
    b) De karakterdata zal verzonden worden door middel van de SPI component.
6. Wanneer alle lijnen op het scherm zijn weergegeven wordt weer gewacht todat de invoer veranderd om nieuwe data weer te geven.

Signaal	**Type**	**Beschrijving**
CLK	Clock	Klokinvoer van 100 MHz.
RST	Input	Synchrone reset waarmee de state terug naar Idle gebracht wordt. De initialisatie zal hierdoor opnieuw worden uitgevoerd.
EN	Input	Via dit signaal wordt deze component geactiveerd.
DATA_IN	Input	20-bit brede datainvoer met die het nummer bevat van de tekstregel die moet weergegeven worden (5-bit per regel).
CS	Output	Chipselect, niet gebruikt op het ZedBoard.
SDO	Output	Seriële data uitvoer, input van OLED.
SCLK	Output	Seriële klok uitvoer, input van OLED.
DC	Output	Data/Command Control.
FIN	Output	Deze vlag geeft aan dat de tekst op het scherm geschreven is.
READY	Output	Deze vlag geeft aan dat de tekst op het scherm geschreven is.

4.4.3 Delay

Zie hoofdstuk 4.3 op bladzijde 11.

4.4.4 SPI control

Deze component verzorgt de SPI communicatie met de OLED display. Aangezien dit de enige component is die de SPI gebruikt is er slechts een beperkte functionaliteit geïmplementeerd. Zo kan er enkel gestuurd worden, en wordt geen CS voorzien aangezien deze aan massa verbonden is op het ZedBoard. De communicatie verloopt volgens een statemachine waarvan de flowchart terug te vinden is in appendix B.7.

Signaal	**Type**	**Beschrijving**
CLK	Clock	Klokinvoer van 100 MHz.
RST	Input	Synchrone reset van de component.
SPI_EN	Input	Via dit signaal wordt deze component geactiveerd.
SPI_DATA	Input	1 byte data die via de SPI-interface verstuurd moet worden.
CS	Output	Chipselect voor SPI, niet gebruikt op het ZedBoard.
SDO	Output	Seriële data uitvoer, input van OLED.
SCLK	Output	Seriële klok uitvoer, input van OLED.
SPI_FIN	Output	Deze vlag geeft aan dat de data verstuurd is.

4.4.5 Gebruikte IPs

CharLib: (Block memory generator) Dit geheugen bevat de data over elk karakter in de ASCII tabel. Deze data wordt verzonden via SPI naar het OLED scherm.

AddressMultiplier: (Multiplier) Deze component zorgt ervoor dat adressen op een snelle manier vermenigvuldigd worden met een vaste offset van 16. Dit is nodig om het juiste startadres te berekenen van een tekstlijn in het geheugen.

TextMemory: (Block memory generator) Dit geheugen bevat 32 lijnen tekst van elk 16 karakters breed (1 karakter is 8 bits) die kunnen worden weergegeven op het OLED scherm.

4.5 VGA HDMI

Deze component overkoepelt alles wat het genereren en tonen van het beeld aangaat. Het bevat vooral verbindingen tussen onderlinge blokken. Het staat ook in voor een nette verbinding naar buiten toe. De klok en data uit RAM komen binnen en het adres voor de RAM en videosignalen verlaten deze component. Het zorgt ook voor de 75MHz pixel klok. Dit wordt uit een 100MHz klok gemaakt met behulp van een PLL en een deler.

Signaal	**Type**	**Beschrijving**
clk_100	Clock	100 MHz klok invoer.
button_dotmode	Input	Signaal voor de selectie van lijnmodus.
button_edgemode	Input	Signaal voor de selectie van randkleuring.
button_theme	Input	Signaal voor de selectie van gradiënt.
button_gridmode	Input	Signaal voor de selectie van blokmodus.
button_averaging	Input	Signaal voor de selectie van uitmiddeling.
button_rounding	Input	Signaal voor de selectie van afronding van de blokjes.
button_peakmode	Input	Signaal voor de selectie van piekdetectie.
data_data	Input	Data bus naar de RAM van de FFT.
oled_data	Input	Serieel data signaal voor het OLED scherm.
hdmi_sda	Input	Data signaal voor I^2C-communicatie.
data_address	Output	Adres bus naar de RAM van de FFT.
vga_r	Output	Rood kanaal voor het VGA signaal.
vga_g	Output	Groen kanaal voor het VGA signaal.
vga_b	Output	Blauw kanaal voor het VGA signaal.
vga_hs	Output	Horizontale synchronisatie puls voor het VGA signaal.
vga_vs	Output	Verticale synchronisatie puls voor het VGA signaal.
hdmi_clk	Output	Klok voor het HDMMI signaal.
hdmi_hsync	Output	Horizontale synchronisatie puls voor het HDMMI signaal.
hdmi_vsync	Output	Verticale synchronisatie puls voor het HDMMI signaal.
hdmi_d	Output	Data kanaal voor het HDMI signaal.
hdmi_de	Output	Data enable voor HDMI signaal.
hdmi_scl	Output	Klok signaal voor I^2C-communicatie.

4.5.1 VGA generator

Deze component staat in voor het daadwerkelijk invullen van de kleuren voor iedere pixel in VGA formaat. Deze kleurwaardes worden real time opgehaald uit een ROM geheugen. Het houdt ook de huidige x- en y-positie van het scherm bij. Door de huidige x-positie te gebruiken als adres voor de datamapping ROM verkrijgen we het adres om de RAM met FFT resultaten uit te lezen. Deze waarde wordt dan vergeleken met de huidige y-positie. Afhankelijk daarvan zal beslist worden om de achtergrondkleur of een gradiëntkleur te tekenen.

ROM Data mapping: ROM geheugen om de 512 frequentiebanden uit de FFT te mappen op de 1280 horizontale pixels van het scherm. Het heeft 1280 geldige adressen en de uitgang is een getal van 0 tot en met 511.

ROM Jet gradiënt: ROM geheugen die de kleurinformatie bevat voor de jet gradiënt. Deze array heeft plaats voor 360 waardes die 24-bit breed zijn.

ROM Oranje gradiënt: ROM geheugen die de kleurinformatie bevat voor de oranje gradiënt. Deze array heeft plaats voor 360 waardes die 24-bit breed zijn.

Signaal	**Type**	**Beschrijving**
clk	Clock	75 MHz klokinvoer.
button_dotmode	Input	Signaal voor de selectie van lijnmodus.
button_edgemode	Input	Signaal voor de selectie van randkleuring.
button_theme	Input	Signaal voor de selectie van gradiënt.
button_gridmode	Input	Signaal voor de selectie van blokmodus.
button_averaging	Input	Signaal voor de selectie van uitmiddeling.
button_rounding	Input	Signaal voor de selectie van afronding van de blokjes.
button_peakmode	Input	Signaal voor de selectie van piekdetectie.
data_data	Input	Data bus naar de RAM van de FFT.
r	Output	Rood kanaal voor het VGA signaal.
g	Output	Groen kanaal voor het VGA signaal.
b	Output	Blauw kanaal voor het VGA signaal.
de	Output	Data enable voor HDMI signaal.
hsync	Output	Horizontale synchronisatie puls voor het VGA en HDMI signaal.
vsync	Output	Verticale synchronisatie puls voor het VGA en HDMI signaal.
data_address	Output	Adres bus naar de RAM van de FFT.

4.5.2 Convert 444-422

Deze component verzorgt de ontdubbeling van de data. Dit wil zeggen dat 1 pixel nu in 2 klokken verstuurd wordt. Dit is nodig voor de HDMI codec. Deze verwacht de pixel telkens in 2 klokcycli. Daarom woden de RGB signalen uitgemiddeld over 2 pixels en in tweevoud aan de kleurconversie aangeboden. Dit laat toe om de Cb en Cr waarde uit te reken per 2 pixels en de Y waarde per pixel. Dit is ook het formaat dat de HDMI codec verwacht.

Signaal	**Type**	**Beschrijving**
clk	Clock	75 MHz klokinvoer.
r_in	Input	Huidig rood kanaal.
g_in	Input	Huidig groen kanaal.
b_in	Input	Huidig blauw kanaal.
hsync_in	Input	Horizontale synchronisatie puls voor het HDMI signaal.
vsync_in	Input	Verticale synchronisatie puls voor het HDMI signaal.
de_in	Input	Data enable voor HDMI signaal.
r1_out	Output	Rood kanaal 1.
g1_out	Output	Groen kanaal 1.
b1_out	Output	Blauw kanaal 1.
r2_out	Output	Rood kanaal 2.
g2_out	Output	Groen kanaal 2.
b2_out	Output	Blauw kanaal 2.
pair_start_out	Output	Synchronisatie puls start pixelpaar.
hsync_out	Output	Horizontale synchronisatie puls voor het HDMI signaal.
vsync_out	Output	Verticale synchronisatie puls voor het HDMI signaal.
de_out	Output	Data enable voor HDMI signaal.

4.5.3 Colour space conversion

Deze component zorgt voor een correcte conversie tussen de RGB en YCbCr kleurenruimte. Omdat het omzetten van RGB kleuren naar YCbCr kleuren vermenigvuldigingen vergt, is deze bewerking verplicht uit te voeren op DSP slices. Daardoor is deze omzetting erg snel en vrij compact in hardware.

Signaal	**Type**	**Beschrijving**
clk	Clock	75 MHz klokinvoer.
r1_in	Input	Rood kanaal 1.
g1_in	Input	Groen kanaal 1.
b1_in	Input	Blauw kanaal 1.
r2_in	Input	Rood kanaal 2.
g2_in	Input	Groen kanaal 2.
b2_in	Input	Blauw kanaal 2.
pair_start_in	Input	Synchronisatie puls start pixelpaar.
hsync_in	Input	Horizontale synchronisatie puls voor het HDMI signaal.
vsync_in	Input	Verticale synchronisatie puls voor het HDMI signaal.
de_in	Input	Data enable voor HDMI signaal.
y_out	Output	Luminicentie kanaal per pixel.
c_out	Output	Chrominentie kanalen afwisselend per 2 pixels.
de_out	Output	Data enable voor HDMI signaal.
hsync_out	Output	Horizontale synchronisatie puls voor het HDMI signaal.
vsync_out	Output	Verticale synchronisatie puls voor het HDMI signaal.

4.5.4 HDMI uitgang

Deze component verzorgt het configureren en aansturen van de HDMI codec. Bij het opstarten wordt eerst een configuratieroutine aangeroepen. Hierna zorgt deze blok voor de verbinding van de videosignalen die van de kleurconversie komen naar de HDMI codec.

Signaal	Type	Beschrijving
clk	Clock	75 MHz klokinvoer.
clk90	Clock	75 MHz klokinvoer 90°in fase verschoven.
y	Input	Luminicentiekanaal voor HDMI.
c	Input	Chrominentiekanaal voor HDMI.
hsync_in	Input	Horizontale synchronisatie puls voor het HDMI signaal.
vsync_in	Input	Verticale synchronisatie puls voor het HDMI signaal.
de_in	Input	Data enable voor HDMI signaal.
oled_data_in	Input	Serieel data signaal voor het OLED scherm.
hdmi_sda	Input	Data signaal voor I^2C-communicatie.
hdmi_clk	Output	Klok voor het HDMI signaal.
hdmi_hsync	Output	Horizontale synchronisatie puls voor het HDMI signaal.
hdmi_vsync	Output	Verticale synchronisatie puls voor het HDMI signaal.
hdmi_d	Output	Data kanaal voor het HDMI signaal.
hdmi_de	Output	Data enable voor HDMI signaal.
hdmi_scl	Output	Klok signaal voor I^2C-communicatie.

I^2C sender

Deze component zorgt voor een correcte initialisatie van de HDMI codec. De configuratie gebeurt door middel van I^2C-communicatie. Eerst en vooral moeten we de codec aanzetten. Vervolgens beschijven we enkele resisters om de codec duidelijk te maken hoe we de data zullen aangeven. Eenmaal alles geconfigureerd is, volstaat het om er correcte videosignalen naar toe te sturen. De codec doet de rest.

Signaal	Type	Beschrijving
clk	Clock	75 MHz klokinvoer.
resend	Input	Controle signaal voor correct ontvangst.
siod	I/O	Data signaal voor I^2C-communicatie.
sioc	Output	Klok signaal voor I^2C-communicatie.

Hoofdstuk 5

Besluit

5.1 Sampelen van audio

Bij het bouwen van een audio spectrum analyzer wordt al snel duidelijk dat je ergens een audiosignaal zal moeten samplen en op één of ander manier zal moeten inlezen. Om audio binnen te lezen werd een codec ter beschikking gesteld. Deze codec werd vervolgens uitgebreid zodat het ook mogelijk werd om audio terug uit te sturen. De moeilijkheden die hierbij ondervonden werden waren vornamelijk te wijten aan de beperkte data die beschikbaar was in de datasheet. Door middel van het programma Sigma Studio was het vervolgens wel relatief eenvoudig om de juiste registerinstellingen te maken.

5.2 Dataverwerking en FFT

Om een spectrum te krijgen van een gesampled signaal moet je er mee rekenen. En een bijkomende moeilijkheid was de vereiste dat de schakeling real time moet werken. Dus met wat we toen al wisten moesten we een algoritme voor een fourier transformatie implementeren en het moest ook snel genoeg zijn om real time te zijn. Daar konden we ons beroepen op de IP library van Xilinx. Deze bevat een FFT blok, die welliswaar via AXI-bus aan te spreken is. De moeilijkheden bij de implementatie van de FFT blok beperkten zicht voornamelijk tot het juist krijgen van de timings van de verschillende signalen.

5.3 Spectrum visualiseren

Om het scherm aan te sturen waren er ook wat moeilijkheden. Op het eerste maakte het niet uit of we dan wel niet HDMI zouden gebruiken. VGA zou makkelijker zijn volgens zekere bronnen. Uiteindelijk zijn we met HDMI verder gegaan. Toch wel met dank aan een stuk voorbeeldcode gevonden op een forum zijn we dan toch aan de slag kunnen gaan. Eens we ons eerste blok op het scherm konden tekenen ging het steeds vlotter. Hoe meer we de code begrepen hoe meer we de code gingen verbouwen. Tot dat het dan weer totaal misliep, om dan een stap terug te zetten en het nog eens te proberen.

5.4 OLED display

Het ontwerpen van de HDMI uitvoer en het OLED scherm gebeurde in twee verschillende projecten. Bij het samenvoegenvan deze projecten werd opgemerkt dat de datapin van het OLED signaal ook verbonden was met een datapin van de HDMI codec. Aanvankelijk werd er geopteerd om ofwel de data te multiplexen. De seriële kloklijnen zouden dan om de beurt op 0 geschakeld worden. Uitijndelijk bleek dat de pin die verbonden was met de HDMI codec steeds op 0 stond en niet actief gebruikt werd. Deze pin maakt deel ui van een 16 bit brede data-interface waarvan enkel de 8 laatste bits in gebruik waren. De gemeenschappelijk pin bevond zich in de eerste 8, ongebruikt pinnen. Een test wees verovolgens uit dat deze data mocht overschreven worden zonder dat dit een invloed gaf op de HDMI uitvoer. Wel moest de OLED data geplaatst worden op een ODDR block omdat de HDMI pinnen via dit principe werkten.

5.5 VHDL toolbox

Aan de hand van een kleine toolbox die ontwikkeld werd door middel van C# gemaakt werd was het op een snelle manier mogelijk om data te genereren voor de verschillende geheugens. Zo wordt onder meer tekstdata voor het OLED scherm automatisch gegenereerd en passend gemaakt voor het geheugen. Ook voor het gradiëntgeheugen kon data gegenereerd worden op basis van een afbeelding.

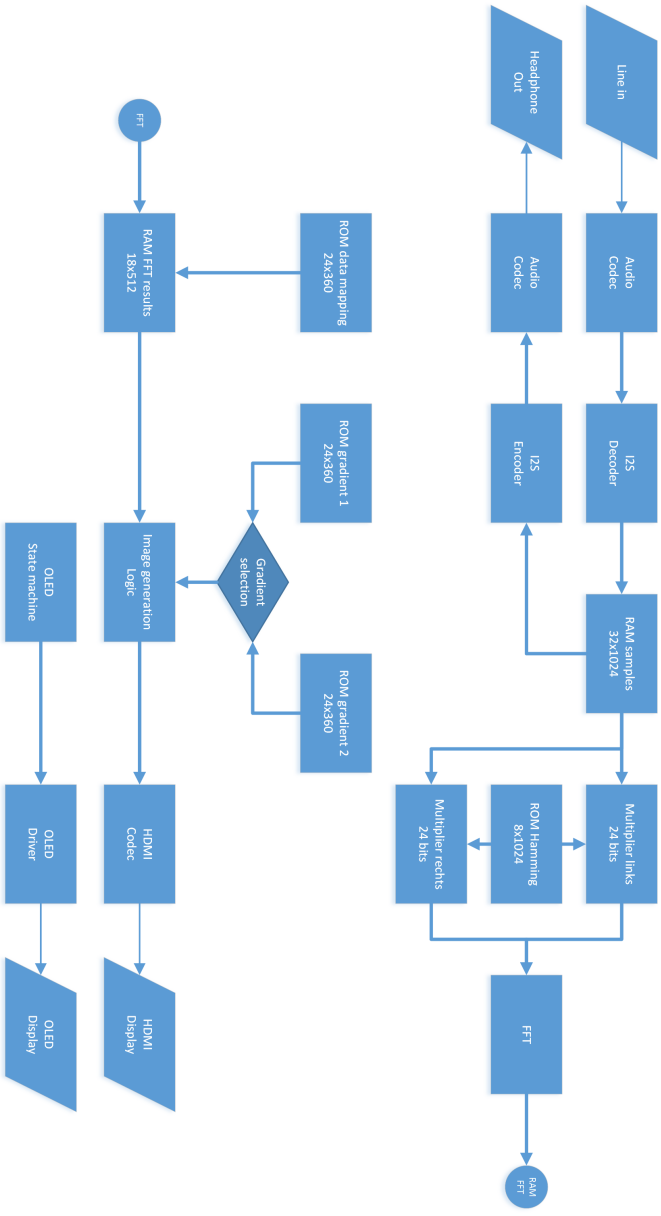
5.6 Algemeen

Algemeen kunnen we stellen dat het een leerzaam project was. Gaande van niks kunnen op een fysisch FPGA development board tot een werkende audio spectrum analyzer was een vrij steile leercurve. Maar eens het eerste gelukt is volgt de rest vrij vlot. Het was vooral een kwestie van die knop om te zetten in ons hoofd. Een tweede moeilijkheid was de omgeving. Op het eerst een overdonderende indruk, een chaos aan features die naderhand uiterst praktisch bleken te zijn. We kunnen zeker stellen dat in de eerste plaats het doel, de opdracht voldaan is. Zelfs met hier een daar een uitbreiding. En in een tweede plaats heeft het zeker en vast een deur wagenwijd opengezet om zelf te gaan experimenteren met de kracht van hardware.

Bijlagen

Bijlage A

Overzicht spectrum analyzer

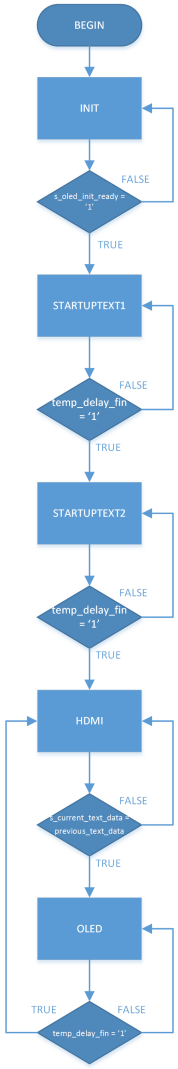


Figuur A.1: Globaal overzicht van de spectrum analyzer

Bijlage B

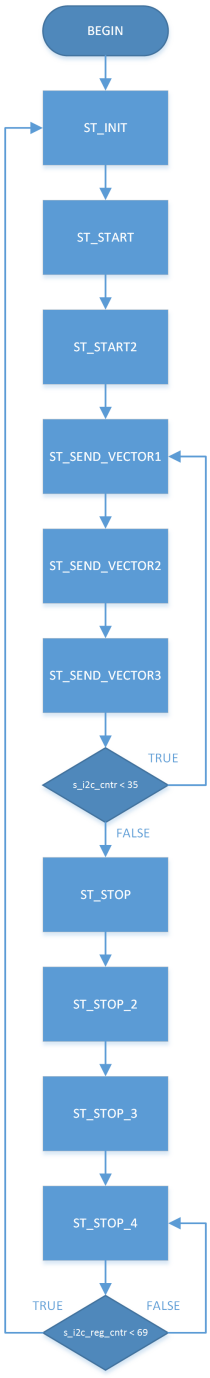
Componenten

B.1 Top



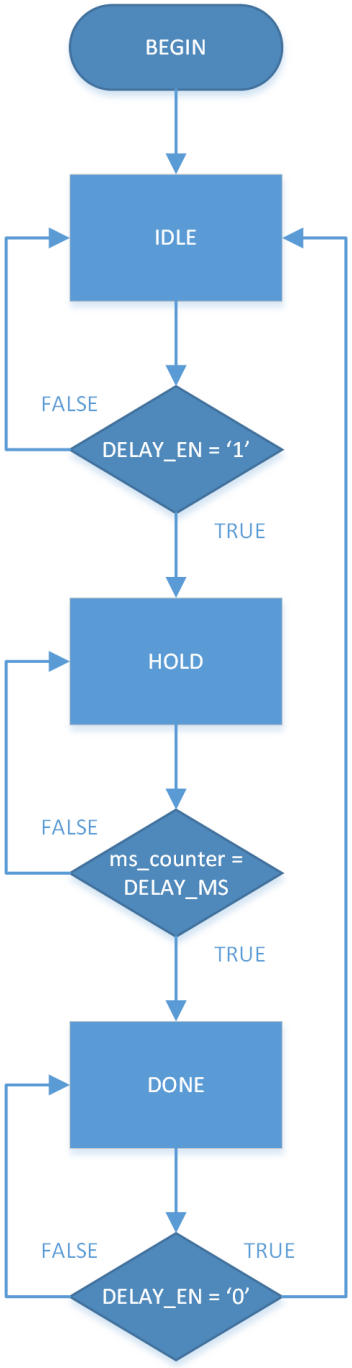
Figuur B.1: Flowchart van top

B.2 Audio interface



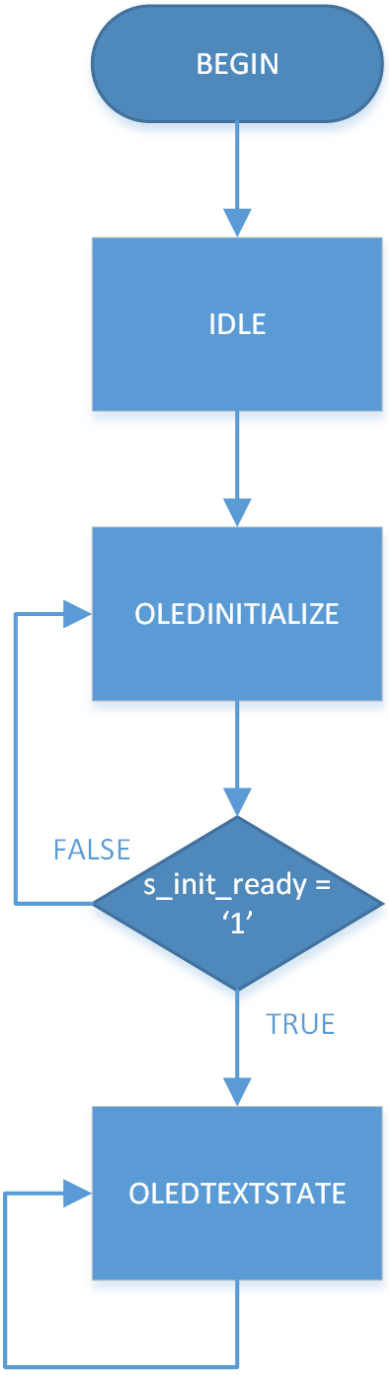
Figuur B.2: Flowchart van de audio interface

B.3 Delay



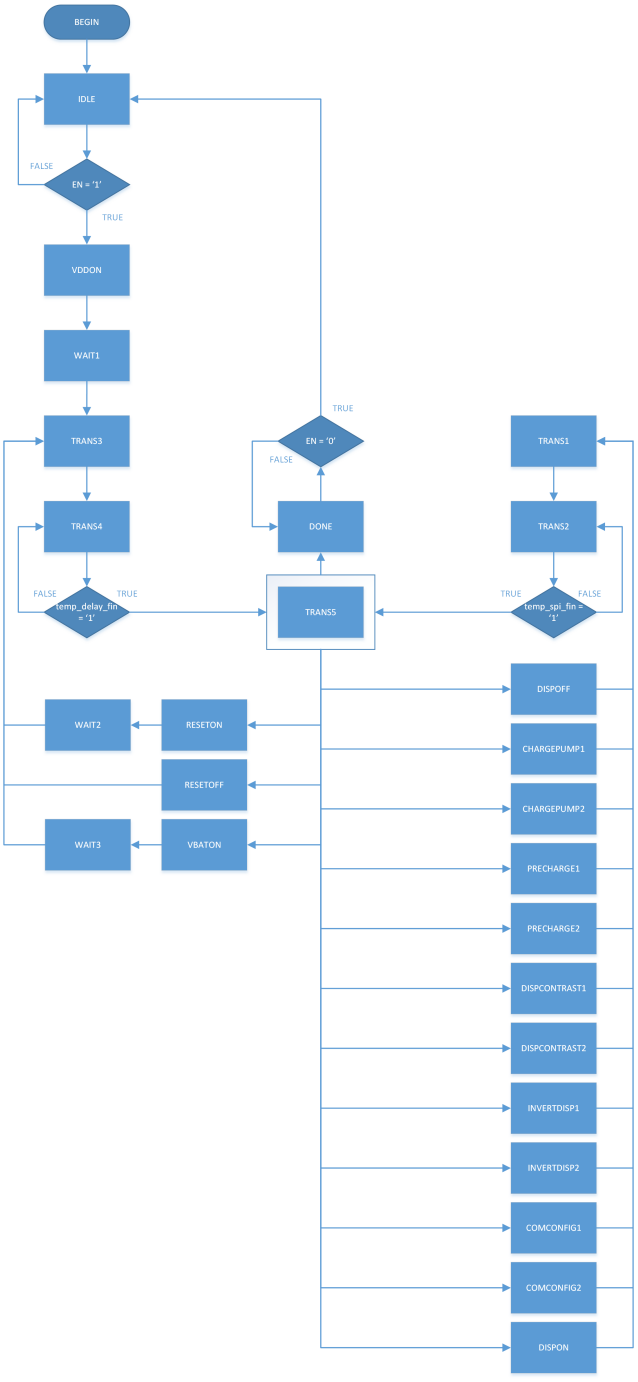
Figuur B.3: Flowchart van de audio interface

B.4 Oled top



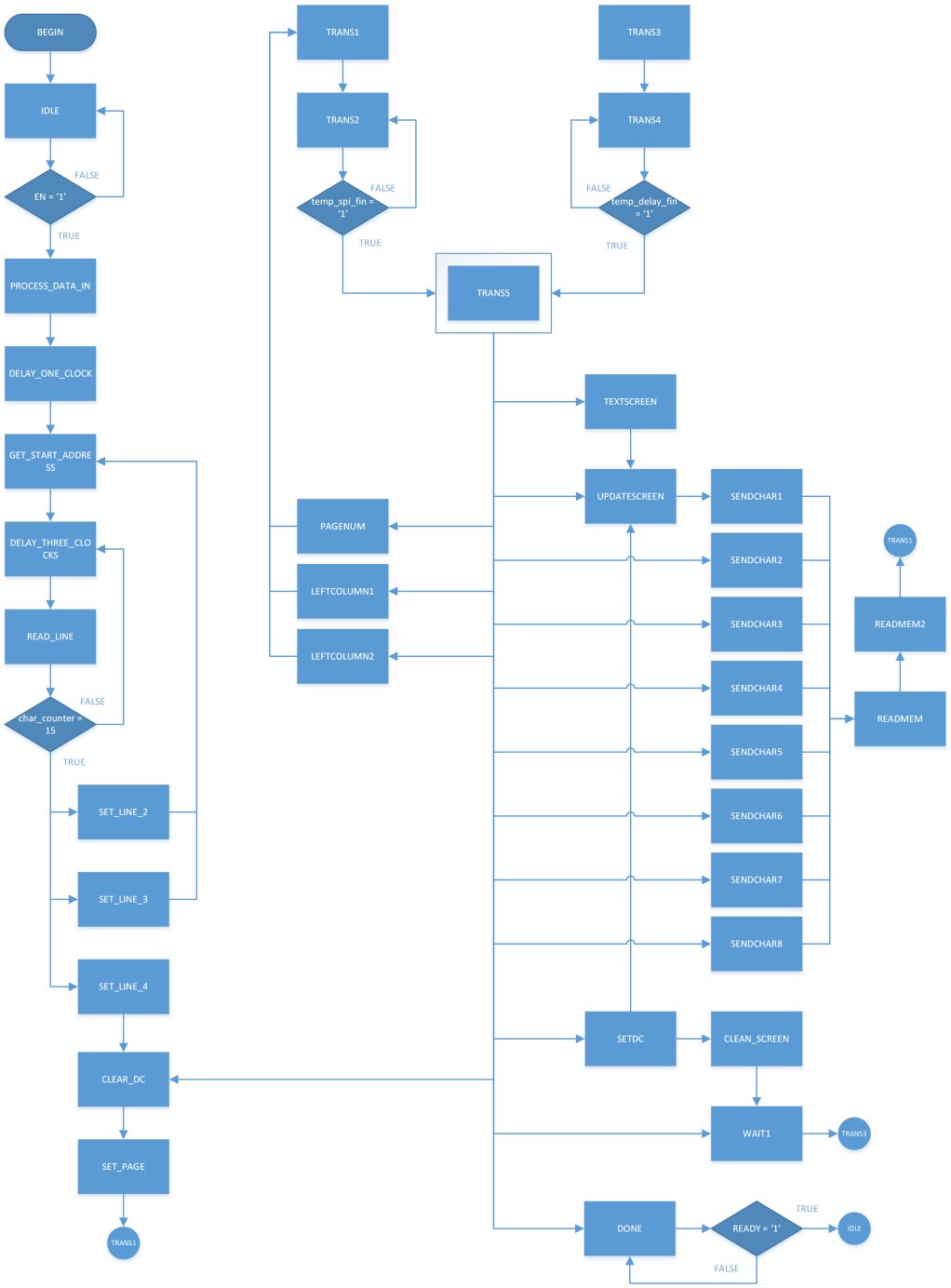
Figuur B.4: Flowchart van oled_top

B.5 Oled initialize



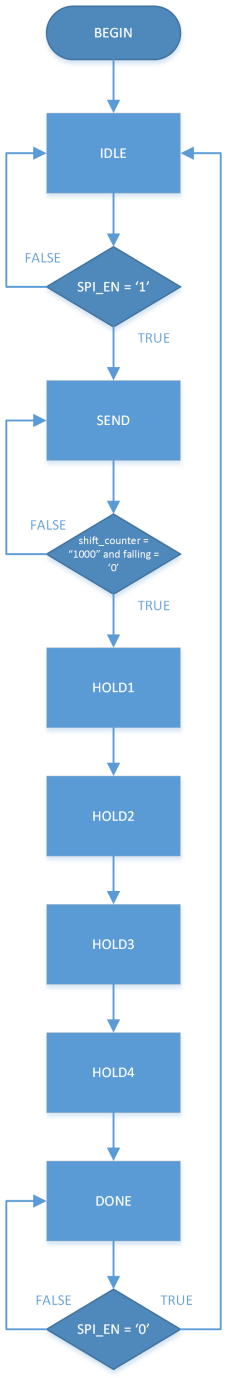
Figuur B.5: Flowchart van OledInit

B.6 Oled text



Figuur B.6: Flowchart van OledText

B.7 SPI control



Figuur B.7: Flowchart van SpiCtrl

Bijlage C

ADAU1761 Settings

Register	Value
0x4000	0x01
0x4001	0x00
0x4008	0x00
0x4009	0x00
0x400A	0x01
0x400B	0x05
0x400C	0x01
0x400D	0x05
0x400E	0x00
0x400F	0x00
0x4010	0x00
0x4011	0x00
0x4012	0x00
0x4013	0x00
0x4014	0x00
0x4015	0x00
0x4016	0x00
0x4017	0x00
0x4018	0x00
0x4019	0x13
0x401A	0x00
0x401B	0x00
0x401C	0x21
0x401D	0x00
0x401E	0x41
0x401F	0x00
0x4020	0x00
0x4021	0x00
0x4022	0x01
0x4023	0xE7
0x4024	0xE7
0x4025	0xE7
0x4026	0xE7
0x4027	0xE7
0x4028	0x00

Register	Value
0x4029	0x03
0x402A	0x03
0x402B	0x00
0x402C	0x00
0x402D	0xAA
0x402F	0xAA
0x4030	0x00
0x4031	0x08
0x4036	0x03
0x40C0	0x7F
0x40C1	0x7F
0x40C2	0x7F
0x40C3	0x7F
0x40C4	0x01
0x40C6	0x00
0x40C7	0x00
0x40C8	0x00
0x40C9	0x00
0x40D0	0x00
0x40D1	0x00
0x40D2	0x00
0x40D3	0x00
0x40D4	0x00
0x40E9	0x10
0x40EA	0x00
0x40EB	0x7F
0x40F2	0x01
0x40F3	0x01
0x40F4	0x00
0x40F5	0x00
0x40F6	0x00
0x40F7	0x00
0x40F8	0x00
0x40F9	0x7F
0x40FA	0x03

Tabel C.1: ADAU1761 settings