

Verslag spectrogram

Academiejaar 2019 – 2020

Robin Nollet, Sebastian Vantomme, Ine Vanderhaeghe

Verslag: bouwen van een spectrogram op een FPGA met behulp van de VHDL-taal

Inhoudsopgave

1	Probleemstelling: een audiospectrogram	3
1.1	Wat is een spectrogram?	3
1.2	Hoe wordt een spectrogram voorgesteld?	3
1.3	Gebruikte methoden om een spectrogram te implementeren	4
1.4	Afwerkingsgraad	4
2	Oplossingsstructuur	5
2.1	Audio inlezen	5
2.2	Fourier nemen	6
2.3	Grafisch weergeven	6
3	Implementatie	6
3.1	Audio inlezen	6
3.2	Fourier nemen	7
3.3	Grafisch weergeven	7
4	Besluit	7

1 Probleemstelling: een audiospectrogram

Doorheen het semester hebben we ons met ons groepje bezig gehouden met het bouwen van een audiospectrogram.

1.1 Wat is een spectrogram?

Het kan handig zijn om snel de frequentieinhoud van een signaal te zien, naar analyse toe. Gezien een signaal ook vaak veranderlijk is in de tijd, kan ook de frequentieinhoud veranderen in functie van de tijd. Er bestaat een methode om de frequentieinhoud van een signaal weer te geven in functie van de tijd. Dit heet een *Spectrogram*. Een spectrogram kan goed samengevat worden volgens de volgende definitie: het is een visuele representatie van de energie in elke frequentie uitgezet in de tijd. Een spectrogram heeft verschillende toepassingen. Enkele voorbeelden hiervan is bv. in de audio, of in de analyse van licht. Dit project focust specifiek op het analyseren van audiosignalen, hoorbaar voor het menselijk oor. Hiervoor nemen we standaard een range van 20Hz tot 20kHz . In de audiowereld worden deze frequenties vaak de *pitch* genoemd van het signaal.

Mogelijke toepassingen van een spectrogram die hoorbare audiosignalen worden hieronder opgesomt:

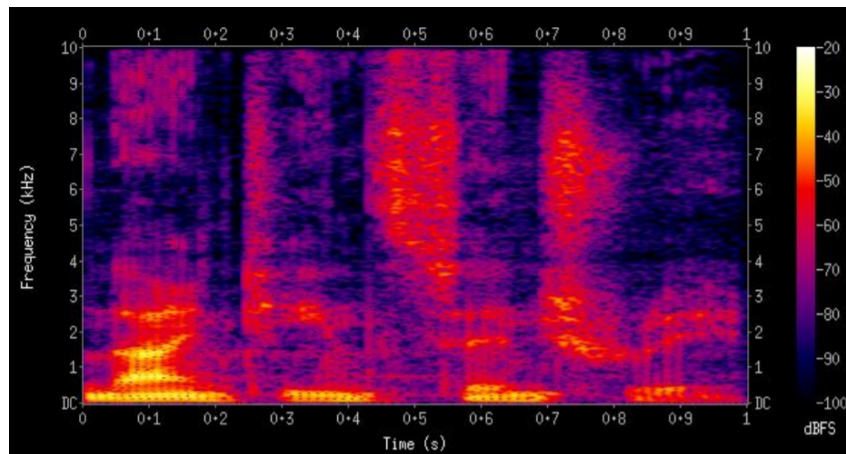
- analyse van Muziek(-instrumenten)
- analyse van Spraak
- analyse van Dierengeluiden
- analyse van RF modulatie technieken

1.2 Hoe wordt een spectrogram voorgesteld?

De grafische voorstelling van een spectrogram zou eigenlijk in 3 dimensies moeten gegeven worden. Dit is echter wat lastig om dan op één oogopslag informatie te kunnen uit aflezen. Daarom worden in de literatuur vaak de volgende afspraken gevolgd:

- De tijd wordt op de horizontale as uitgezet;
- De frequentie wordt op de verticale as uitgezet;
- De sterkte van de frequentie wordt vaak weergegeven met behulp van kleurintensiteit of een kleurgradiënt.

In figuur 1 is een typische grafische voorstelling gegeven.



Figuur 1: Een typische grafische voorstelling van een spectrogram

1.3 Gebruikte methoden om een spectrogram te implementeren

In de literatuur kunnen 2 veelgebruikte methoden voor de implementatie van een Spectrogram gevonden worden.

1. Met behulp van de FFT: van het ingenomen signaal wordt de *Fast Fourier Transform (FFT)* genomen, wat een versnelde versie is van de *Discrete Fourier Transform (DFT)*. Dit toont de frequentieinhoud van het signaal. Een belangrijke eigenschap van deze transformatie is dat ze een discrete ingang (de audio) en een discrete uitgang (de frequentieinhoud van dit signaal) heeft.
2. Met behulp van een filterbank: hier wordt een reeks banddoorlaatfilters geïmplementeerd, die allemaal op hetzelfde signaal werken. Door na iedere filter dan te gaan kijken wat de energie van het signaal is, kan dit als een maat gebruikt worden voor de frequentieinhoud van ons ingangssignaal.

Naar implementatie toe heeft men dan een keuze, dit hangt af welke resolutie men wilt in de frequentie. Wil men een grote resolutie in de frequentie, gaat men eerder kiezen voor de aanpak met de FFT. Als men met een lage frequentieresolutie zich tevreden kan nemen, wordt er vaak geopteerd voor de implementatie met die meerdere filters.

Wij hebben hier gekozen voor de implementatie met een FFT, omdat deze een stuk flexibeler is in implementatie.

1.4 Afwerkingsgraad

Onze opdracht bestaat er uit om zelf een real-time spectrogram te maken van een audiosignaal dat aangeboden wordt door een FPGA. Dit spectrogram moet gevisualiseerd worden op een extern scherm via VGA of HDMI.

Als eerste moet er dus een analoog audiosignaal kunnen binnengelezen worden via de audiocodec chip. Deze zorgt zelf voor de digitalisatie van het signaal en geeft dit door aan de FPGA. Er moet dus communicatie mogelijk zijn tussen de codec chip en de FPGA.

Daarna moet het spectrum bepaald kunnen worden. Er worden een aantal samples genomen op het gedigitaliseerde audiosignaal. Daarop wordt een FFT uitgevoerd. Hierbij kan gewerkt worden met aansluitende of overlappende blokken van de samples. Als met overlapping gewerkt wordt zou de resolutie fijner zijn, maar het zou ook complexer zijn.

Ten laatste moet de verzamelde data na de FFT op een 2D map komen die het spectrogram voorstelt. Eens het volledige scherm gevuld is, moet de oudste meting verwijderd worden en schuift het spectrogram een tijdseenheid op via een FIFO.

In het volledige systeem moet rekening gehouden worden met de timing en mogelijke asynchrone signalen.

Er zal tijdens de evaluatie van het project met verschillende testsignalen gecontroleerd worden als het project juist ontworpen is. Voorbeelden van testsignalen zijn FM modulatie, frequentie sweep, blok golf, DTMF signalen, ...

Er zijn enkele minimum vereisten in dit project:

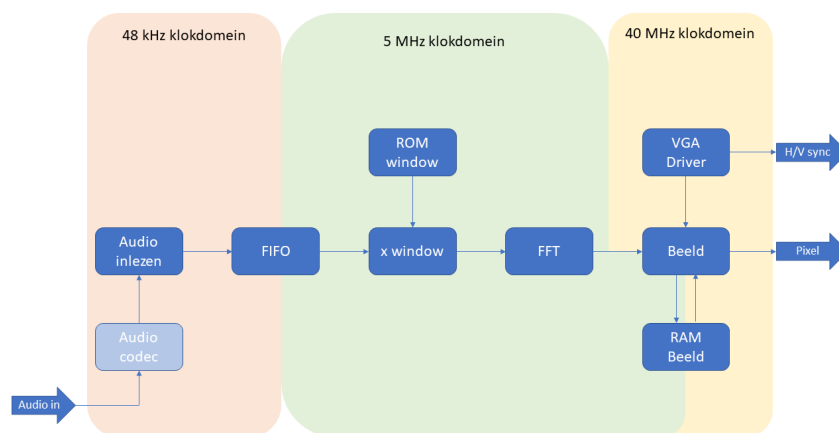
- Er wordt gewerkt met een FPGA bord naar keuze (Zedboard, Zybo, PYNQ, ...)
- Interface met audiocodec chip
- Real-time FFT op audio input
 - 1 kanaal (mono)
 - Aansluitende blokken van N samples
- Opstellen en updaten van spectrogram
- VGA/HDMI interface logica
- Beschikbaar stellen van spectrogram op een extern beeldscherm

Daarnaast zijn nog enkele mogelijkheden tot uitbreiding:

- Stereo spectrogram
- FFT op overlappende tijdssignalen
- 3D spectrogram
 - Amplitude als kleurintensiteit
- Full HD
- Audio loopback
 - Signaal terug naar buiten via line-out of headphone

2 Oplossingsstructuur

Voor dergelijke probleemstelling is de eerste stap om een *pipeline* op te stellen, een stappenplan van wat er met de data moet gebeuren van input tot aan output. Bij deze opdracht hebben we de pipeline toegepast gevonden in figuur 2.



Figuur 2: toegepaste pipeline

Zoals te zien valt op de figuur is het werk grotendeels op te splitsen in 3 grote delen, namelijk aan de hand van het klokdomein. Er hoeft dan gewoon via een dual-port-RAM een overschakeling te worden gemaakt tussen deze verschillende klokdomeinen. We hebben het werk opgesplitst in de volgende delen:

1. Audio inlezen. Kort samengevat is dit het inlezen van de audio via de audiocodec en stockeren in de FIFO.
2. Fourier nemen. Dit is de data uit de FIFO halen, er een window op toepassen en er daarna de fouriertransformatie op loslaten. De output van de FFT wordt dan doorgegeven aan Beeld.
3. Grafisch weergeven. Hier wordt de data gegeven van de FFT opgeslagen in een RAM, die dan ook wordt uitgelezen aan een andere kloksnelheid. Zo wordt de data visueel op het scherm gerepresenteerd.

2.1 Audio inlezen

Om de audio binnen te lezen, zijn er 2 belangrijke bestanden: *adau1761* en *audio_if*. Deze bestanden lezen de data binnen en genereren daar de juiste clock signalen voor. In het top level wordt daar dan een audio loopback van gemaakt om de data die binnengelezen werd ook weer naar buiten te sturen. De data die binnengelezen werd wordt ook naar een FIFO gestuurd, zodat deze op de juiste momenten binnengelezen kan worden in de FFT controller.

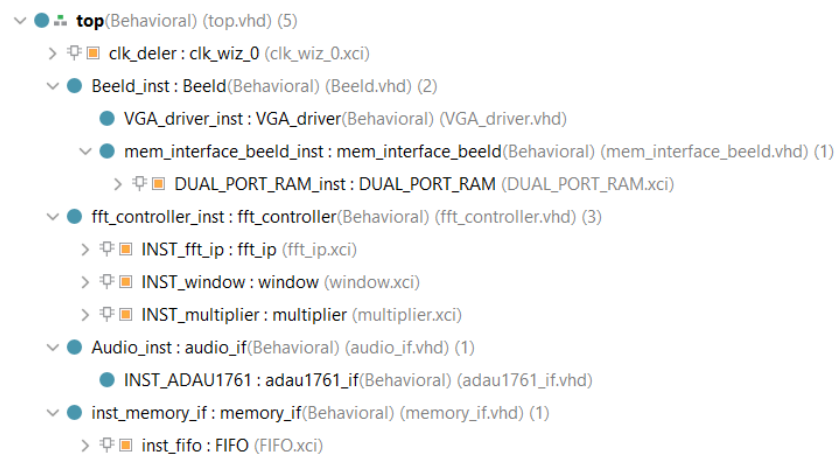
2.2 Fourier nemen

2.3 Grafisch weergeven

Om de berekende resultaten dan ook grafisch weer te geven, hebben we gebruik gemaakt van VGA. Om dit wat beter beheersbaar te houden is de architectuur opgesplitst in *VGA_Driver* en *mem_interface_Beeld*. *VGA_Driver* heeft als functie om Hsync en Vsync te genereren. Daarnaast genereert het ook een X en Y, wat de huidige positie is van de pixel die moet worden doorgestuurd, tesamen met een flag die hoog komt als men in het actieve gebied zit van het beeld (dus niet in de porch of syncpulses).

mem_interface_Beeld zorgt met de gegeven X en Y waarde dat de juiste pixel naar buiten komt. Ook zorgt het voor de overschakeling tussen de verschillende klokdomeneinen.

3 Implementatie



Figuur 3: Hierarchie van de implementatie

Zoals te zien is op deze figuur is er een top laag die alle andere onderdelen met elkaar verbindt. Hier wordt een klok deler gemaakt, zodat elk onderdeel in zijn eigen klokdomenein kan werken.

Voor het beeld is een VGA driver nodig en een memory interface. Dit is een dual port ram, omdat de data die er in geschreven wordt niet op dezelfde klok werkt als de data die er uit gelezen wordt. Het schrijven gebeurt door de FFT controller, het lezen door de beeld interface.

De FFT controller maakt gebruik van 3 IP's: een om de FFT te berekenen, een die het window bevat die moet vermenigvuldigd worden met de waarden die uit de FFT komen en een die de andere 2 vermenigvuldigd met elkaar.

De audio interface maakt gebruik van een driver voor de *adau1761* chip die op de FPGA aanwezig is.

Daarnaast is ook een memory interface. Deze maakt gebruik van een FIFO die op 2 verschillende klokken werkt. Hier wordt de data doorgegeven tussen de audio interface en de FFT controller. De audio interface zal data in de FIFO schrijven volgens een klok die ze zelf genereert, de FFT controller leest ze uit wanneer het geheugen vol is op een klok die sneller is dan die van de audio interface.

3.1 Audio inlezen

Er kan gebruik gemaakt worden van de audiocodec die aanwezig is op het Zedboard. Dit is de *adau1761*. Hiervoor moest een driver geschreven worden. Voor ons project hebben we een driver kunnen gebruiken die we gekregen hadden van Dhr. Verslype. Deze driver krijgt zowel serieel als parallelle data binnen.

De samples van het linker en rechter kanaal zijn 24 bit breed. Deze worden verwerkt in de driver en er wordt een linker en rechter output gegenereerd die 24 bit breed zijn en een seriële data output.

Daarnaast moest een interface geschreven worden die de clock genereert en de data die de driver binnenleest op een juiste manier door geeft naar het top level. Ook hier hebben we gebruik gemaakt van een interface die we gekregen hadden van Dhr. Verslype. Deze interface heeft een kloksignaal nodig van 100 MHz en 12,288 MHz. Deze genereren 3 nieuwe kloksignalen: de *master clock*, de *digital audio bit clock* en de *digital audio left-right clock*. Deze worden gebruikt in de adau1761 codec.

De configuratie van de audio interface werkt met het I^2C protocol, het doorgeven van audio data gebeurt via het I^2S protocol.

In het top level wordt daarna een audio loopback gemaakt. De data wordt ook doorgegeven aan een FIFO memory, zodat deze uitgelezen kan worden door de FFT controller. Het geheugen is 2048 waarden groot, dus wanneer deze vol is wordt een flag hoog gezet zodat de FFT controller weet dat alle waarden die in de memory staan geldig zijn. Het adres waarop de data geschreven moet worden in de FIFO wordt gegenereerd door middel van een counter tussen 0 en 2048.

3.2 Fourier nemen

3.3 Grafisch weergeven

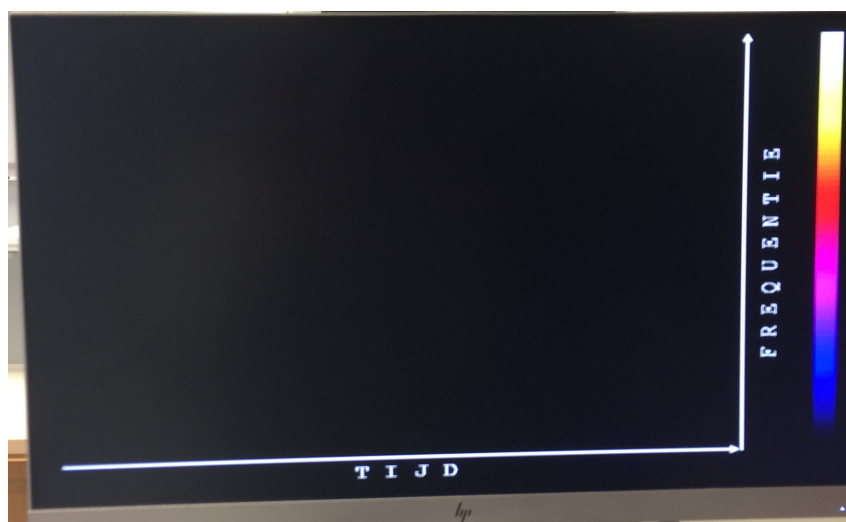
4 Besluit

Het is ons gelukt om een spectrogram op een extern scherm af te beelden die real-time de data verwerkt. Alle minimum vereisten zijn voldaan. We hebben gekozen om met een Zedboard te werken en om het scherm aan te sturen via VGA. Er is een interface met de audiocodec chip en de FFT wordt berekend op real-time waarden.

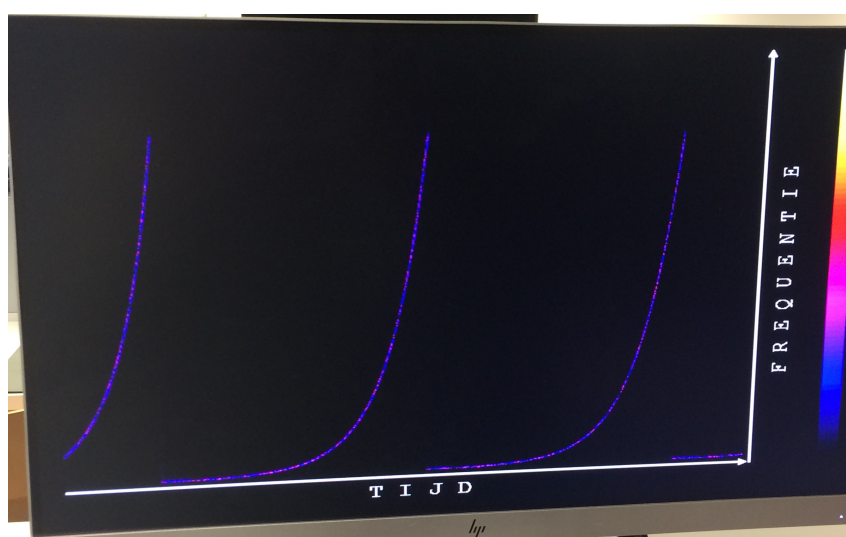
Er zijn ook enkele uitbreidingen die we hebben kunnen implementeren: het spectrogram is 3D (de amplitude van de data wordt aangegeven met kleuren) en er is een audio loopback geïmplementeerd. De audio wordt als stereo binnengelezen, maar als mono signaal doorgegeven aan de FFT controller. Dit project zou dus nog kunnen uitgebreid worden door er een stereo spectrogram van te maken. Verder zou de FFT ook kunnen berekend worden op overlappende tijdssignalen en het scherm zou kunnen uitgebreid worden naar full HD.

In figuur 4 staat het scherm van het spectrogram die wij gemaakt hebben wanneer er geen geluid aan de audiocodec gegeven wordt.

In figuur 5 staat het scherm wanneer een chirp signaal aan de audiocodec aangeboden wordt.



Figuur 4: Spectrogram



Figuur 5: Spectrogram met chirp signaal