Opdrachten Lab 2 – PSK en frequentiesynchronisatie

Leerdoelen

Aan het einde van dit lab ben je in staat om:

- Het IQ-Diagram uit te leggen;
- Een PSK signaal te synchroniseren in frequentie;
- Een Root Raised Cosine filter toe te passen;
- Data te ontvangen met een PSK signaal.

Voorbereiding op de les

- Lees de hoofdstukken uit de voorbereiding

Opdrachten

3.1 Eerst zullen we het IQ-Diagram beter gaan bekijken. We beginnen bij een ideale situatie: Het PSK signaal is ontvangen, gedemoduleerd, we hebben alleen bits. Deze bits simuleren we met een *Vector Source* met de vector `(-1, 1)`. Het stuurt dus continu -1,1,-1,1, etc. Vanaf dit punt gaan we het uitbreiden naar een meer reële situatie door telkens een nadelig kanaaleffect toe te voegen. Voer de volgende stappen uit, en probeer bij elke opdracht te begrijpen wat je ziet in de QT GUI *Constellation Sink*. Als je er niet uitkomt, vraag dan de docent om uitleg of voer de Python codes van het boek uit. Hier worden dezelfde stappen genomen in Python maar dan voor QPSK.

A Voeg een QT GUI *Constellation Sink* toe en bekijk de constellatie. Wat zie je? Probeer de vector source wat andere waarden laten uitsturen, bijvoorbeeld `(1+1j, 1)`. Wat zie je nu? Zet de vector source weer terug naar `(-1, 1)`.

B In de echte situatie is er ruis in de amplitude en fase. Eerst simuleren we alleen faseruis. Dit ontstaat in echte situaties door imperfecties van de oscillatoren die worden gebruikt. Voeg een *Phase Noise Generator* toe tussen de vector source en de *Constellation Sink*. Geef de *Phase Noise Generator* een Alpha van 1 en een Noise Magnitude van 0.2. Bekijk de constellatie. Wat zie je?

C Naast faseruis is er ook algemene ruis in fase en amplitude. Dit kan bijvoorbeeld ontstaan door de versterkers en ADCs. Voeg aan de uitgang van het faseruisblok ook normale ruis toe door een *Noise Source* en *Add* blok te gebruiken. Wat zie je? Probeer de ruis wat te verminderen door de waarde van de noise source te verlagen. Wat zie je nu? Een ruisamplitude van ongeveer 100 mV is een mooie waarde. De waarde definieert de standaarddeviatie van de ruis (dus 1σ).

D Tot nu toe hebben we een perfecte frequentiesynchronisatie gesimuleerd, want de data zit rond de 0 Hz. Voeg een frequentieverschuiving toe op de uitgang van het add blok, met behulp van een *Signal Source* en *Multiply* blok. Bekijk de constellatie na de frequentieverschuiving. Wat zie je? Probeer de waarde van de *Signal Source* te veranderen. Wat zie je nu? Een waarde van 0,01 Hz is een mooie waarde om het effect goed te zien.

E Als laatste is het niet reëel dat de vector source een amplitude heeft van 1. Voeg een *Multiply Const* blok toe tussen de vector source en het faseruisblok, en verlaag de waarde naar bijvoorbeeld 0.3. Wat zie je?

F Maak gebruik van een *Virtual Source* en *Virtual Sink* blok om de gesimuleerde data door te koppelen. Geef het een ID van `signaal_met_afwijking` en koppel de uitgang van het multiply blok aan de sink. Zet de source klaar voor de volgende opdracht.

3.2 Zender en ontvanger kunnen tot 22 kHz afwijken in frequentie en zelf nog fluctueren over de tijd. Dit betekent dat de ontvanger de zender actief zal moeten volgen. We kunnen dit volgen opsplitsen in grove- en fijne synchronisatie. In eerste instantie zullen we de grove synchronisatie handmatig doen met een slider. De fijne synchronisatie doen we met behulp van een *Costas Loop*. Voer de volgende stappen uit:

A Voeg aan het schema van opdracht 3.1 een *Costas Loop* toe zoals is uitgelegd in de presentatie. Tot welke frequentieafwijkingen corrigeert dit blok de frequentie-afwijkingen? Dus hoe nauwkeurig moet de grove synchronisatie zijn?

3.3 De docent zal tijdens dit lab twee PSK signalen uitzenden. Het linker kanaal wordt differentieel verstuurd, het andere kanaal niet. De tekst die wordt verstuurd is “ELE!”. Bij elke opdracht is het de bedoeling dat je die tekst kunt laten verschijnen op jouw computer. De benodigde delay in de bitreeks zal voor iedereen en iedere keer anders kunnen zijn. PSK signalen zijn te demoduleren met een productdetector, net als bij DSB-SC. Alleen nu hebben we de hulp van de Costas loop! De docent stuurt de signalen uit rond de 433,5 MHz. De signalen hebben een bitrate van 1kbps. Het RRC filter heeft een rolloff (α) van 1 gebruikt. Begin een nieuw schema en voer de volgende stappen uit:

A Zoek het rechter kanaal op met de Pluto en filter enige andere signalen zo goed mogelijk weg door een *Root Raised Cosine Filter* te gebruiken. Hiermee gebruik je een matched filter en voer je handmatig de grove frequentiesynchronisatie uit.

B De amplitude van het signaal zal behoorlijk klein zijn. Voeg een *AGC* toe om het signaal naar een bruikbaar niveau te brengen.

C Gebruik een *Costas Loop* om het signaal exact op 0 Hz te zetten en bekijk of dit werkt in het constellatiediagram. Je zou een lijn op de reële as moeten krijgen, het zijn nog geen bits.

D Zet de uitgang van de Costas Loop om naar bits. Gezien de uitgang van de Costas loop de data zoveel mogelijk reëel heeft gemaakt, kun je het reële deel van de

data direct door een *Binary Slicer* halen en een keep 1 in N blok. Bekijk de de bits in een constellatie sink. Je krijgt ongeveer het figuur van opdracht 3.3.

E Laat de verzonden tekst op je computer zien via dezelfde methode als bij opdracht 2.1.

F Je zult zien dat de data soms niet goed wordt ontvangen. Dit komt door de besproken problemen in 4.8. Differentiële Codering. Het linker kanaal is differentieel verstuurd. Pas een *Differential Decoder* toe op de symbolen (dus voor het Pack K Bits blok) en stem af op het andere kanaal. Bekijk of de data nu wel consistent goed wordt ontvangen.

3.4 Nu zullen we een *FLL Band-Edge* blok gaan gebruiken voor hulp bij de grove synchronisatie. De pluto heeft een afwijking van 25ppm, of ±11 kHz voor een draaggolf van 433,5 MHz. Hierdoor kunnen we de correctie niet na het matched filter uitvoeren als ons filter, bij een datarate van 1kbps, slechts een bandbreedte heeft van 2 kHz. Voeg het *FLL Band-Edge* blok toe aan het schema van opdracht 3.3 direct voor het RRC filter, om te helpen het signaal netjes in het matched filter te krijgen.