Teil I

Grundlagenteil - Alle

Der erste Teil dieses (letzen) Übungszettels (Abgabe und Ausarbeitung wie gewohnt) legt die Grundsteine für die Beendigung des Endprojektes. Er ist, im Gegensatz zum zweiten Teil wo eine Trennung in *RX*- und *TX*-Gruppen erfolgt, von allen Teams zu absolvieren.

Direct Digital Synthesis (DDS) erlaubt die Erzeugung eines *beliebigen* (meist periodischen) digitalen Signals unter Verwendung einer Signalwerttabelle (lookup table). Diese Übung bildet die Basis zur Konzeptentwicklung für die Signalerzeugung im Endprojekt von SCD 5.

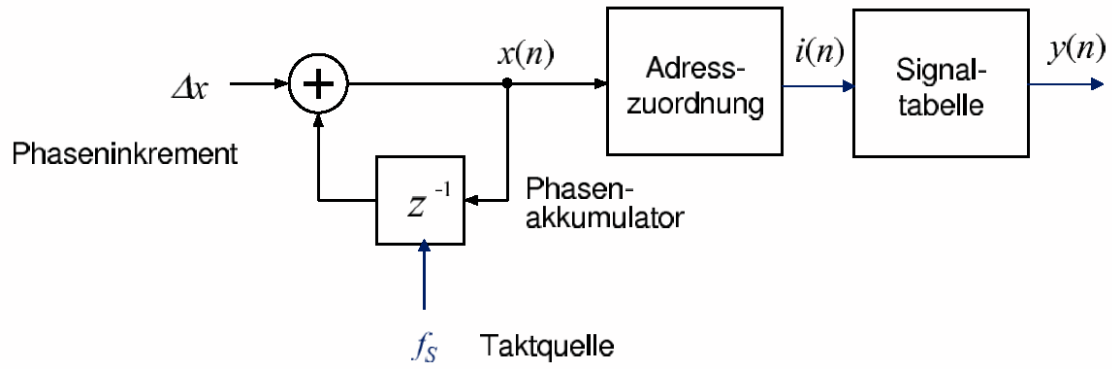

Abbildung 1: Blockschaltbild für DDS

1. Aufgabe *Realisierbare Frequenzen mit DDS*

Schreiben Sie ein Matlab Skript, dass bei einer gegebenen Bitbreite des Phasenakkumulators alle realisierbaren Frequenzen ausgibt. Beachten Sie: Für den Phaseninkrement sind nur positive Werte sinnvoll. Tipp: Die Formel mit der Sie aus einem Phaseninkrement und der Abtastfrequenz eine DDS Frequenz berechnen können, finden Sie im Skriptum.
Beispielausgabe des Skripts:

```
>> frealizable(3,44117)
Ermitteln der realisierbaren Frequenzen bei 3 Bit
mit fs = 44117 Hz:
Phaseninkrement:     Frequenz
0.250000             5514.625000
0.500000             11029.250000
0.750000             16543.875000
```

Geben Sie zusätzlich zum Quellcodes Ihres Skripts die Ausgabe für 5 und 6 Bit bei einer Abtastfrequenz von $44117 Hz$ ab.

2. Aufgabe *DDS mit Gleitkommazahlen*

Abbildung 1 zeigt ein Blockschaltbild zur Realisierung von DDS. Die Funktion dieses Blockschaltbildes

ist im Skript *dds.m* realisiert. Machen Sie sich mit der Funktion dieses Skriptes vertraut. Beachten Sie, dass der Phasenincrement als *fractional* Wert erwartet, und intern in eine Tabellenadresse umgerechnet wird.
Führen Sie mit Hilfe von *dds.m* folgende Simulation durch. Vervollständigen Sie dafür das Skript *Ue4_2_start.m*.

- Erzeugen Sie ein Sinussignal der Frequenz 14 kHz, falls möglich. Sollte diese Frequenz mit den andern Parametern dieser Aufgabe nicht exakt zu realisieren sein, so erzeugen Sie ein Sinussignal mit der am nächsten liegenden realisierbaren Frequenz. Geben Sie diese Frequenz und die Abweichung zur geforderten Frequenz in Prozent an.
- Verwenden Sie dazu als Abtastfrequenz 44117 Hz und eine Phasenakkumulatorwortbreite von 4 Bit.
- Wählen Sie dazu eine geeignete Anzahl von Signalwerten die durch DDS erzeugt werden, um eine sinnvolle Darstellung eines mit *DFT* erstellten Spektrums zu gewährleisten. (Alternativer Denkansatz: Wieviele Perioden eines Sinussignals mit der realisierbaren Frequenz müssen sie mit fs abtasten um eine Periode der Abtastwertfolge zu erhalten?)
- Stellen Sie das mittels DDS erzeugte Zeitsignal dem optimalen Sinussignal (der realisierbaren Frequenz) gegenüber.
- Berechnen Sie das Spektrum und stellen Sie es über der Frequenz dar. Ergibt sich wirklich ein Sinussignal? Messen Sie die Frequenz des Sinussignals. Stimmt diese Frequenz mit der von Ihnen berechnenten (oder durch das Skript von Aufgabe 1 ausgegebenen) überein?

Überlegen Sie welche minimale Anzahl von Abtastwerten Sie in der Tabelle ablegen müssen, damit am Ausgang der DDS ein reines (abgetastetes) Sinussignal ergibt. Wieviele Signalwerte muß Ihre DDS liefern bis sich die Signalwerte wiederholen (eine Periode der Abtastwerte)? Warum ist das mit DFT ermittelte Spektrum von der Anzahl der Signalwerte abhängig mit denen eine DFT berechnet wird?

3. Aufgabe *DDS mit Interpolation*

In dieser Aufgabe soll wieder eine DDS eines Sinussignals realisiert werden. Vervollständigen Sie dazu das Skript *Ue4_3_start.m*.

- Realiseren Sie eine Frequenz die möglichst genau $2757Hz$ entspricht. Messen Sie die von Ihnen erreichte Frequenz und geben Sie diese an.
- Verwenden Sie dazu 8 Tabellenwerte, und einen Phasenincrement von 4 Bit. Erzeugen Sie 5 Perioden der Sinusschwingung. Wie sieht das von Ihnen erwartete Ausgangssignal der DDS aus?
- Betrachten Sie dazu das Spektrum des generierten Signales. Was stellen Sie fest?
- Verwenden Sie zur Interpolation des generierten Signals ein FIR Tiefpassfilter. Wählen Sie dazu geeignete Frequenzen für ω_{Pass} und ω_{Stop}, sowie geignete Rippel in Durchlass- und Sperrbereich. Begründen Ihre Wahl! Achten Sie darauf, dass Ihr Filter auf jeden Fall nicht mehr als 100 Koeffizienten benötigt!
- Geben Sie den maximalen Fehler in Prozent zwischen dem optimalen Sinussignal und dem durch DDS generierten Sinussignals an.
- Stellen Sie nun das Betragsspektrum des gefilterten Signals dar. Erläutern Sie, warum die Spektrallinien an den Punkten der Grundfrequenz ihrer Sinusschwingung nicht exakt den Wert $.5$ aufweisen (falls dies so ist). Hätten Sie etwas anderes erwartet?
- Wie könnte man verhindern, dass für die Erzeugung der geforderten Sinusschwingung ein Tiefpass benötigt wird?

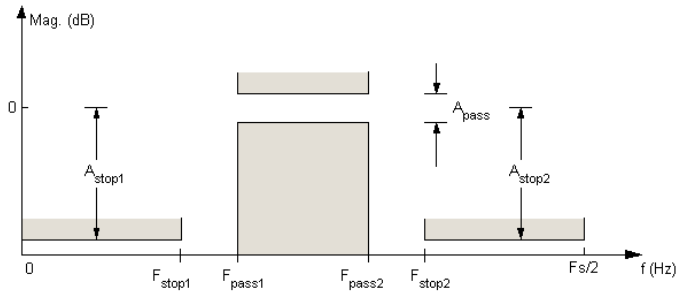

Abbildung 2: Entwurfsparameter eines Bandpassfilters im Frequenzgang

4. Aufgabe *Filterentwurf mit fdatool*

Entwerfen Sie einen quantisierten Bandpassfilter mit *fdatool* nach folgenden Entwurfsparametern (Siehe Abb. 2.

```
Fs        =  48000  Hz
Fstop1    =  7000  Hz
Fpass1    =  8000  Hz
Fpass2    =  9000  Hz
Fstop2    =  10000  Hz
Astop1    =  40  dB
Apass     =  1 dB
Astop2    =  40  dB
```

Verwenden sie den *Equiripple Entwurf* und außerhalb und innerhalb des Filters überall die gleiche Quantisierung. Welche minimale Ordnung des Filters brauchen Sie (unter 255 Koeffizienten !!!)?
Stellen sie den Frequenzgang des quantisierten und unquantisierten Filters dar. Ab welcher Bitbreite wird das Entwurfsschema erfüllt?

Teil II

Abschluss des Endprojektes

Im zweiten Teil der Übung sollen nun die jeweiligen Gruppen (RX oder TX) Simulationsstrecken mit den folgenden Parametern erstellen. In der Übungsstunde werden den Gruppen ihre Frequenzen und damit ihre Partnergruppen zugewiesen.
Beachten Sie: Fehler in der Konzeptentwicklungsphase eines Kommunikationssystems sind die teueresten Fehler im gesamten Entwicklungs- und Fertigungsprozess. Die Korrektur eines solchen Fehlers im Endprodukt ist i.A. viel aufwändiger als z.B.: die Korrektur eines Fehlers in der VHDL Implementierung.
Daher wird empfohlen, die Simulationsstrecken der jeweiligen Partnergruppen zu vereinen um eventuelle konzeptuelle Inkompatibilitäten der spätern FPGA Implementierung so früh wie möglich zu erkennen.

Als wichtiger gemeinsamer Parameter wird eine Länge von 157 Abtastwerten pro übertragenen Bit vereinbart. Das ergibt eine Bitrate von 281 bit/sec bei $f_s = 44117Hz$.

5. Aufgabe *Sender*

Zur Realisierung der Sendesignale sollen zwei Sinussignale mittels DDS erzeugt werden. Als maximale Abweichung von einem idealen Sinussignal ist ein absoluter Fehler von 0.05 zulässig. Vergleichen Sie dazu das von Ihnen generierte Signal mit einem optimalen (mit f_s abgetastetem) Sinussignal. Achten Sie jedoch darauf, die Abweichung so klein als möglich zu halten. Durch eine kleinere Abweichung beim Sender erhöhen Sie die Performance des Sender/Empfänger Gesamtsystems!

- Überlegen Sie wieviele Tabellenwerte Sie benötigen.
- Welche Phaseninkrementbitbreite benötigen Sie?
- Obergrenze für die Darstellung der Tabellenwerte und des Phaseninkrements sind 16 Bit
- Verwenden Sie maximal 1024 Tabellenwerte

Ihr Aufgaben sind nun:

- Erstellen Sie einen Generator zur Erzeugung eines FSK Signals in Abhängigkeit eines binären Eingangssignals
- Eine Beispielfunktion in Matlab hat folgende Signatur:

```
function y = FSKGenerator(fs, f1,f2, nTable,...
                          bIncrement, bTable, nBit, Bitstream)
% y = FSKGenerator(fs, f1,f2, nTable, bIncrement, bTable, nBit, Bitstream)
% fs         Sampling Frequency
% f1         Frequency 1 to generate
% f2         Frequency 2 to generate
% nTable     Number of Values for Table
% bIncrement Bitwidth of phase increment
% bTable     Bitwidth of Table entries
% nBit       Number of samples per bit
% Bitstream  Bits to encode
```

- Achten sie darauf, dass es nicht zu Phasensprüngen im Ausgang des FSK Generators kommt! Tipp: Ändern sie den Phasenincrement der DDS in Abhängigkeit des gesendeten Bits.

Simulieren Sie das Gesamtsystem im *Fixed Point*. Bestimmen Sie nun die Parameter so, dass Sie die geforderte Genauigkeit erreichen. Stellen Sie das generierte Datensignal (qantisierte Sinusschwingungen nach einem zu sendenden Bitstrom) einem idealen (unquantisierten) Sendesignal gegenüber. Plotten sie den Fehler zwischen diesen beiden Signalen. Wie groß ist der maximale Fehler? Achten Sie darauf den Grenzwert auch wirklich einzuhalten!!!

6. Aufgabe *Empfänger*

Ihr zugeordnetes Senderteam hat die Aufgabe die geforderten Sinussignale mit einer maximalen absoluten Abweichung von 0.05 zu erzeugen. Diesen Sachverhalt können Sie sich zu Nutze machen indem Sie ein Quellensignal nach folgendem Skriptfragment erzeugen:

```
% Precision of DDS:
DDSPrecison = 0.05;

% Erzeugung zweier freilaufender Oszillatorschwingungen
t = 0:Ts:N*T_symb-Ts;
NPoints = length(t);
oszillator_1 = cos(2*pi*fu*t)+rand(1,NPoints)*DDSPrecison-DDSPrecison/2;
oszillator_2 = cos(2*pi*fo*t)+rand(1,NPoints)*DDSPrecison-DDSPrecison/2;

% ****************** sender **********************
% Erzeugung einer Anzahl (N) zufälliger Datenbits
data_transmit = round(rand(1,N));


% Erzeugung des Sendesignals
s = [];
```

```
datasignal = [];
for k = 1:N
    if data_transmit(k) == 0
        s = [s, oszillator_1((k-1)*N_symb+1:k*N_symb)];
        datasignal = [datasignal zeros(1,N_symb)];
    else
        s = [s, oszillator_2((k-1)*N_symb+1:k*N_symb)];
        datasignal = [datasignal ones(1,N_symb)];
    end
end
```

Im Gegensatz zum Senderteam verwenden Sie zwei freilaufende Sinusschwingungen. Dabei treten Phasensprünge beim Umtasten zwischen den beiden Sinusschwingungen auf, die im Signal ihres Sendeteams nicht vorhanden sein sollten. Es ist daher zu erwarten, dass ihr Entwurf sich bei der Kombination der Simulationsstrecke ihres Partnerteams besser verhält als mit dieser einfachen Generierung des Sendesignals.
Beim Empfänger wird die Hüllkurvendetektion mittels der Absolutwertbildung (besonders einfach in Hardware zu realisieren) und einem nachgeschalteten Tiefpass realisiert. Auf Grund der Linearität kann der Tiefpass hinter die Addition gesetzt werden. Da die Anforderung an die Tiefpässe im oberen und unteren Zweig gleich sind, kann also ein Tiefpass eingespaart werden.
Somit ergibt sich eine Struktur wie in Abbildung 3 ersichtlich.

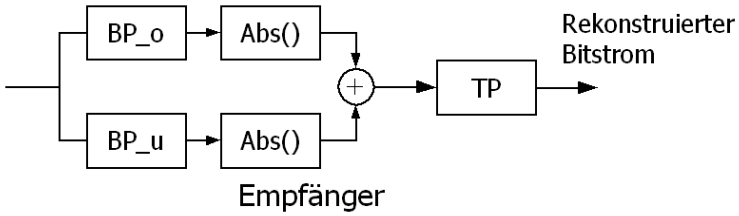

Abbildung 3: Aufbau des Empfängers

Bauen Sie nun eine Simulationsstrecke in Matlab um den Empfänger zu simulieren. Generieren sie dazu ein Signal mit zufälligen Bits und daraus das FSK Quellensignal. Einen AWGN-Kanal können Sie nach folgendem Codefragment modellieren:

```
% ******************* channel ********************
Psignal = 1/length(s)*sum(s.^2);

noise = 2*randn(1,length(s))-1;

Pnoise = 1/length(noise)*sum(noise.^2);

NoiseScale = sqrt(Psignal/10^(snr/10)/Pnoise);

noise = noise * NoiseScale;

r = s + noise;          %r is the signal through the AWGN channel
```

Ihr Aufgabe ist nun:

- Entwerfen Sie passende Bandpass- und Tiefpassfilter.
- Entwerfen Sie die Bandpassfilter derart, dass ein Störsender mit Freqenzen im Abstand von 1000 Hz von ihrer untern bzw. oberen Sendefrequenz die Bitfehlerrate ihres Systems nicht erhöht. Nehmen Sie an, dass dieser Sender mit der gleichen Sendeleistung sendet wie ihr eigener Sender.
- Obergrenze der Bitbreite für Signaldarstellung und Filterarithmetik sind 16 Bit.
- Verwenden Sie maximal 256 Filterkoeffizienten. Skalieren Sie ihre Filter geeignet.
- Simulieren Sie ihre Strecke. Bei einem SNR größer gleich 5dB sollen keine Bitfehler auftreten. Legen Sie Ihr System jedoch so aus, dass Sie mit einem möglichst kleinem SNR auskommen. Zur Simulation der Filter können sie die Funktion: *filtfir_symm_qa.m* verwenden.
- Wieviel dB SNR benötigen Sie damit keine Bitfehler auftreten?
- Simulieren Sie dazu mindestens 10000 Bits bzw. Sendesymbole.
- Tasten Sie das Ausgangssignal Ihres Empfängers im Symboltakt ab und vergleichen Sie das Empfangssignal an diesen Stellen mit den gesendeten Bits.